Departamento de computación
Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Sistema de recomendación basado en comentarios textuales

Estudiante: Pedro Rivero Vilariño
Dirección: María Amparo Alonso Betanzos
Dirección: Verónica Bolón Canedo
Dirección: Carlos Gómez Rodríguez

A Coruña, 7 de septiembre de 2020.

A mis padres, mi hermano y mi pareja

Agradecimientos

Primeramente quiero agradecer a mis tutores Amparo, Carlos y Verónica su guía y su apoyo para la realización del proyecto. Sin sus recomendaciones me habría estancado en muchos puntos del estudio y desarrollo del mismo. También quiero dar las gracias a mi familia y mi pareja Lorena, por el apoyo recibido a lo largo de los años y especialmente estos últimos meses en los que la vida ha sido especialmente dura conmigo. Por último, quiero dar las gracias especialmente a mi amigo Antón, por su amistad a lo largo de la carrera, tanto como compañero de piso como de trabajo y estudio.

Resumen

El proyecto descrito en este documento tiene como objetivo construir un comentario de texto personalizado de cada usuario de un Sistema de Recomendación (RS, por sus siglas en inglés) hacia un producto o servicio que este no haya utilizado, basándose en el total de comentarios textuales de los distintos usuarios utilizados en el RS.

El fin de un RS es sugerir a los usuarios nuevos productos o servicios que puede que no conozcan aún, basados en las preferencias de usuarios con características similares. Desde un punto de vista económico, estos RS son de vital importancia, ya que muchas de las mayores compañías del mundo por capitalización de mercado (como Google, Amazon o Facebook) están substancialmente basadas en plataformas que recomiendan productos a sus usuarios. Habitualmente, los usuarios de un producto o servicio dejan patente su opinión mediante un sistema de puntuación numérico, un comentario en lenguaje natural y, opcionalmente, una serie de fotos. Este proyecto se centra en utilizar los comentarios como forma de evaluar la opinión del usuario y extraer información sobre sus gustos para poder ofrecer recomendaciones más personalizadas.

En concreto, el objetivo de este proyecto será evaluar las opiniones de usuarios de TripAdvisor sobre hoteles para tratar de predecir la reacción de un usuario frente a un hotel determinado que este no haya visitado aún. Para ello, primeramente será necesario obtener un conjunto de datos con opiniones reales de usuarios de TripAdvisor, para posteriormente preprocesar los comentarios para poder trabajar con ellos (eliminación de palabras vacías, puntuación, etc.). Seguidamente se obtendrá una representación densa del lenguaje para poder interpretar los comentarios y se usará una aproximación probabilística junto con Deep Learning para poder predecir las reseñas de usuario/hotel. Por último se implementarán distintas métricas para evaluar los comentarios obtenidos.

Abstract

The project described in this document has the objective of building a personalized textual comment for each user in a Recommender System (RS) to a product or service that the user hasn't used in the past, based on every users' textual comments used in the RS.

The end goal of a RS is to suggest new products or services to users that they may not know about, based on the preferences of users with a similar profile. From an economic point of view, RS are of vital importance, since many of the world's biggest companies by revenue (like Google, Amazon or Facebook) are substantially based on platforms that recommend products to their users. Users of a product or service usually show their opinion by means

of numeric-based ratings, a natural language comment and, optionally, a series of photos. This project is centered around the idea of exploiting the comments as a mean of evaluating user opinions and extracting information about their preferences with the end goal of making personalized decisions.

Specifically, the objective of this project will be to evaluate TripAdvisor's user's opinions on hotels to try and predict the reaction of a user when presented with an hotel he/she has not visited yet. To bring about this project, it will be necessary to obtain a dataset with real reviews of TripAdvisor's users, later preprocess the comments (removal of stop-words, punctuation etc.). Next, a dense representation of the language of the comments will be obtained so as to be able to interpret them and a probabilistic approach will be used in conjunction with Deep Learning to predict the reviews. Lastly, different metrics will be implemented to assess the quality of the obtained reviews.

Palabras clave:	**Keywords:**
Sistema de Recomendación	Recommender System
Reseña	Review
Lenguaje Natural	Natural Language
Modelado de lenguaje	Language modeling
Aprendizaje Profundo	Deep Learning

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En esta sección se da una pequeña introducción a este proyecto y su importancia, se describen los objetivos que se plantearon al inicio del proyecto y se expone la estructura de la memoria del proyecto.

1.1 Contexto

A día de hoy, el uso de internet es ubicuo en todos los países desarrollados, de tal manera que se ha integrado en la rutina y forma de vida de su población. Según las cifras del INE de 2018 [13], internet está presente en el 83.4% de los hogares españoles, un valor que no deja de crecer con el paso del tiempo, debido a la fuerte integración del mismo en la cultura de las nuevas generaciones.

Gracias a la omnipresencia de internet, el comercio online en España ha experimentado en los últimos años un crecimiento exponencial en facturación (ver figura 1.1), de tal manera que en 2018, el gasto online representa un 20% del consumo en España, convirtiéndose en un sector clave de la economía.

De igual manera que en el comercio tradicional, la publicidad ejerce una poderosa influencia en las personas a la hora de comprar productos y servicios, tanto es así, que muchas de las mayores compañías del mundo por capitalización de mercado (como Google, Amazon o Facebook) están substancialmente basadas en plataformas que recomiendan productos a sus usuarios. Estos Sistemas de Recomendación (RS, por sus siglas en inglés) son, por tanto, un sector increíblemente lucrativo, y cualquier mejora de los mismos podría aportar enormes beneficios económicos. Además, estos sistemas aportan un gran valor a los usuarios, al ayudarles a encontrar productos y servicios en línea con sus intereses de forma gratuita.

Puede definirse un RS como un sistema de filtrado de información cuyo objetivo final es

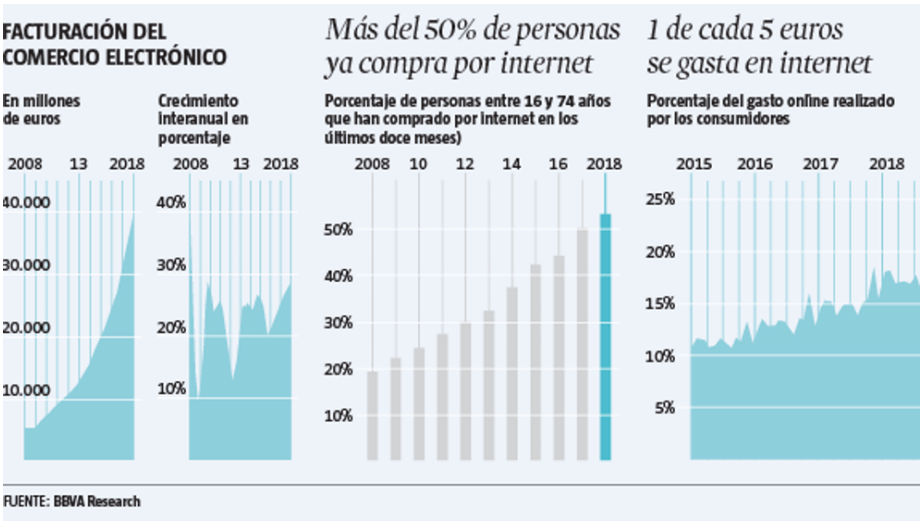

Figura 1.1: Evolución y escala del comercio electrónico en España. [1]

mostrar productos o servicios que el usuario pueda querer consumir. Para ello, los RS hacen uso de diferentes datos de entrada, entre ellos características de los productos (por ejemplo, categorías, como los diversos tipos de productos de un supermercado), características de los usuarios (datos personales como la edad y la localización) y preferencias de los usuarios sobre productos. La recolección de preferencias puede ser explícita (pedir al usuario su opinión sobre un producto que ha adquirido, normalmente consistente en una puntuación numérica, que puede ir acompañada de una reseña de texto y fotografías) o implícita (por ejemplo analizando qué productos ha visto el usuario en una web, aún sin adquirirlos).

Webs como Amazon, Aliexpress o TripAdvisor entre otras permiten a sus usuarios dejar opiniones sobre los productos o servicios que han adquirido. Estas suelen estar compuestas de una puntuación numérica, un comentario/reseña en lenguaje natural (NL, por sus siglas en inglés) y opcionalmente una serie de fotos. En la figura 1.2 puede verse una reseña de un usuario de Amazon a un teléfono móvil, que contiene todos los elementos anteriormente mencionados: puntuación numérica (el número de estrellas coloreadas, 4 sobre 5), un comentario y fotos del producto. También es posible para otros usuarios marcar la opinión como útil para indicar que esta reseña es de especial relevancia.

Existen principalmente tres tipos de RS según su filtrado:

- **Filtrado basado en contenido**: se basan en las descripciones de los productos (una serie de palabras clave) o su categoría y el perfil de preferencias de los usuarios (qué productos prefiere). Con esta información, los RS tratan de recomendar productos similares a otros que le han gustado al usuario en el pasado.

Figura 1.2: Reseña de un usuario de Amazon a un producto.

- **Filtrado colaborativo:** estos métodos de filtrado se basan en encontrar usuarios que compartan preferencias. Si dos o más usuarios tienen muchas preferencias en común, tendrán el mismo gusto. Estos usuarios componen pues un grupo llamado vecindario. Se recomendará a un usuario productos que no haya utilizado con anterioridad pero que hayan sido adquiridos y/o evaluados positivamente por su vecindario. También existe el filtrado colaborativo "item to item", basado en encontrar la relación entre productos en lugar de usuarios.

- **Filtrado híbrido:** combinación de los dos métodos de filtrado anteriores.

En este proyecto se construyó un RS de filtrado colaborativo, centrado en el uso de los textos de reseñas en lenguaje natural como fuente de información para construir un RS. Puede verse en la figura 1.3 el estado inicial de la base de datos de un sistema de recomendación de este tipo, una matriz relacionando usuarios y sus valoraciones sobre productos o servicios.

En literatura, los textos de reseñas se han utilizado principalmente como medio de mejorar las predicciones basadas en el uso de las puntuaciones numéricas de las reseñas [14], [15], [16] (explicado con detalle en la sección 3.1), mediante el uso de métodos simples como Bag-of-words, que consiste en la simplificación de textos en lenguaje natural a un multiset de palabras, obviando la relación gramatical de las palabras. En este proyecto se utiliza un método más avanzado para la extracción de información de comentarios, Doc2vec (explicado con detalle en la sección 5.3), que sí es capaz de explotar la información contenida en el orden de las palabras y la estructura lingüística de los textos, además del contexto y lenguaje del

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	V 1,1			V 1,4		V 1,6
User 2		V 2,2	V 2,3		V 2,5	
User 3		V 3,2				V 3,6
User 4	V 4,1			V 4,4	V 4,5	

Figura 1.3: Esquema inicial de sistema de recomendación.

conjunto de reseñas.

Además, como particularidad del proyecto, se utilizan textos en lenguaje natural como resultado último de la recomendación del RS. Existen pocos antecedentes en literatura del uso de textos como recomendación, ya sean conjuntos de palabras clave [17] o textos generados por algoritmos [18]. Sin embargo, en este proyecto se garantiza la coherencia y estructura de los textos al recomendar textos ya vistos en el entrenamiento del algoritmo de RS.

En resumen: este proyecto tiene como factor diferencial el mejor aprovechamiento de la información contenida en los comentarios en lenguaje natural que realizan los usuarios de RS, así como el uso de textos coherentes en lenguaje natural como resultado último de la recomendación.

1.2 Objetivos

Este proyecto se centra en utilizar los comentarios de texto que usuarios dejan sobre productos/servicios como forma de evaluar sus opiniones y extraer así información sobre sus gustos para poder ofrecer recomendaciones personalizadas en forma de textos en lenguaje natural.

Para ello se hizo uso de un entorno de aplicación concreto, utilizando las opiniones de usuarios de TripAdvisor sobre hoteles para tratar de predecir la reacción de un usuario concreto frente a un hotel que no haya visitado todavía. Primeramente fue necesario obtener un conjunto de datos con opiniones reales de usuarios de TripAdvisor, para posteriormente preprocesar la base de datos y los comentarios de texto para poder trabajar con ellos (eliminación de palabras vacías, puntuación etc.). Seguidamente se obtuvo una representación densa

del lenguaje mediante el uso del algoritmo Doc2vec para poder interpretar los comentarios en NL. Esta representación densa del lenguaje se refiere a que se relacionará cada comentario con un vector de baja dimensionalidad, lo cual se explica con detenimiento en la sección 5.3.1. Posteriormente se usó una aproximación probabilística junto con deep learning para poder predecir las valoraciones usuario/hotel en forma de vector, para después transformar dicho vector a un texto en lenguaje natural. Por último se implementaron distintas métricas para evaluar los comentarios obtenidos.

En la figura 1.4 puede verse todo este proceso en un diagrama.

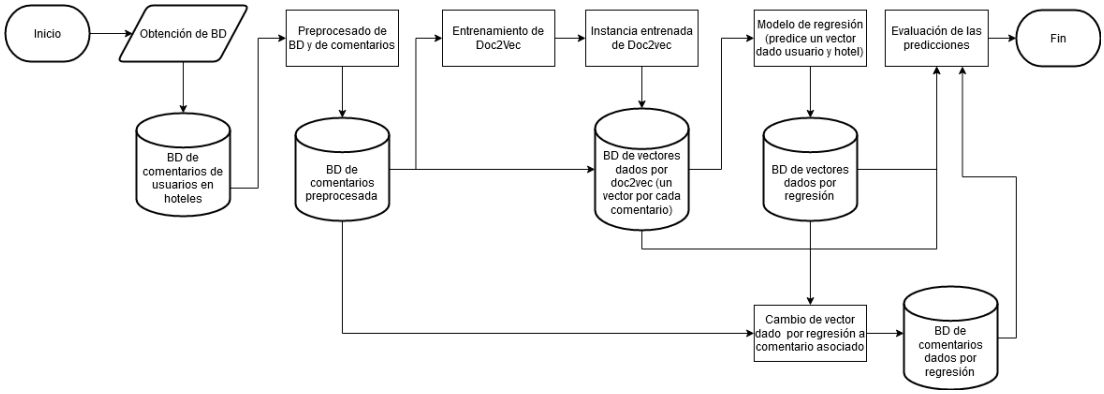

Figura 1.4: Diagrama del flujo del proyecto.

1.3 Estructura de la memoria

A continuación se expone la composición de la memoria, enumerando y describiendo de manera simple el contenido de cada uno de los capítulos que la conforman.

- **Capítulo 2 - Planificación:** se explica la metodología usada en el proyecto, su planificación temporal, su seguimiento y la estimación de costes del mismo.
- **Capítulo 3 - Estado del arte:** se analiza el estado del arte de los RS en relación al uso de texto como elemento nuclear del análisis. Además se hace un pequeño análisis de varios de las grandes RS utilizadas en la actualidad.
- **Capítulo 4 - Tecnologías:** se introducen brevemente las diversas tecnologías empleadas a lo largo del desarrollo del proyecto, incluyendo el lenguaje de programación usado, distintas bibliotecas notables en el proyecto y los entornos de desarrollo utilizados, así como el entorno de ejecución del proyecto.
- **Capítulo 5 - Materiales y métodos:** se expone y analiza el único material base utilizado durante el proyecto, una base de datos de la web TripAdvisor, así como los diversos

métodos estudiados para su aplicación en el proyecto.

- **Capítulo 6 - Desarrollo:** se describe cada uno de los pasos llevados a cabo para conseguir los resultados finales del proyecto a partir de la base de datos inicial, aplicando algoritmos descritos en el capítulo "Materiales y métodos" y evaluando su rendimiento. Se comentan los resultados intermedios de cada paso.
- **Capítulo 7 - Resultados finales y análisis:** se exponen y analizan los resultados finales obtenidos tras el desarrollo del proyecto.
- **Capítulo 8 - Conclusiones y líneas futuras:** se exponen las conclusiones obtenidas después de la realización del proyecto junto con posibles mejoras que se podrían aplicar al mismo.

Capítulo 2
Planificación

En este capítulo se explica la metodología usada en el proyecto, su planificación temporal, su seguimiento y la estimación de costes del mismo.

2.1 Metodología Scrum

La metodología utilizada en este proyecto es una metodología ágil basada en el marco de desarrollo Scrum [19], aunque, debido a que el proyecto es un trabajo de fin de grado y debido a la disponibilidad de los tutores del mismo, se han hecho ciertas adaptaciones. Este marco de desarrollo es ideal para proyectos de investigación, por su naturaleza flexible e iterativa.

Scrum es un marco de desarrollo que contiene una serie de buenas prácticas para trabajar en equipo y obtener el mejor resultado posible en los proyectos. Este marco está fundamentado en la teoría del empirismo, la cual afirma que el conocimiento viene de la experiencia y basa la toma de decisiones en el conocimiento factual. Se caracteriza por ser un desarrollo incremental e iterativo en una serie de pasos llamados Sprints, al final de cada cual se obtiene un producto completo, y se planifica el Sprint siguiente basándose en el resultado del anterior. Existen tres pilares que soportan cada implementación del control de procesos empírico: transparencia, inspección y adaptación.

- **Transparencia:** Los aspectos más fundamentales del proceso deben ser visibles para los responsables del resultado. Para facilitarlo, estos aspectos deben estar definidos en un estándar común a todo el equipo.
- **Inspección:** Los usarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum (ver sección 2.1.2) para detectar variaciones no deseadas.
- **Adaptación:** Si un inspector determina que existen aspectos de un proceso que se desvían de unos límites aceptables, el proceso o materiales se ajustarán con la mayor presteza posible para reencauzarlo en la trayectoria deseada.

2.1.1 Roles en Scrum

El Equipo Scrum se divide en tres roles: Product Owner, Scrum Master (también conocido como facilitador) y Equipo de desarrollo.

- **Product Owner:** Es la persona que representa a los clientes y sus expectativas, siendo el responsable en última instancia de maximizar el valor del producto final. Es el responsable de establecer los requisitos del sistema y gestionar la lista de tareas a realizar.
- **Equipo de desarrollo:** Equipo de profesionales que trabajan en el desarrollo de cada Sprint. Este equipo se organiza a sí mismo y funciona como una unidad, sin subdivisiones.
- **Scrum Master:** Esta persona es un intermediario responsable de gestionar el proceso Scrum, mediante el soporte y comunicación con los roles anteriormente nombrados, así como la organización a la que pertenecen.

2.1.2 Documentos

Durante la aplicación de Scrum se generan una serie de documentos, diseñados para maximizar la transparencia de información clave entre los miembros del equipo. Son tres documentos: el Product Backlog, los Sprint Backlog y los incrementos.

- **Product Backlog:** Este documento gestionado por el Product Owner es una lista ordenada de todo lo necesario para el proyecto. Se listan las características, funciones, requerimientos, mejoras y arreglos que constituyen los cambios que se realizarán en el producto en futuros Sprints. Este documento se caracteriza por su evolución constante durante el desarrollo del proyecto.
- **Sprint Backlog:** Este documento gestionado por el Equipo de Desarrollo contiene un subconjunto de requisitos del Product Backlog seleccionados durante un Sprint, además de una planificación de cómo se implementarán dichos requisitos. Por lo general estos requisitos se subdividen en tareas de corta duración (menor de 16 horas) que los miembros del equipo escogen según les parezca conveniente.
- **Incremento:** Cada incremento es la suma de todos los elementos completados del Product Backlog durante un Sprint, integrada con el trabajo de los Sprints previos. Al final de cada Sprint, el incremento debe estar completo y ser funcional y utilizable. Los incrementos son gestionados por el Equipo de Desarrollo.

2.1.3 Eventos

Los eventos se utilizan en Scrum para crear una regularidad y minimizar las reuniones no definidas en este marco. Cada evento tiene una duración máxima adherida, para no desperdiciar tiempo. Se trata de los Sprints y cuatro tipos de reuniones alrededor del mismo:

- **Sprint:** Este es el núcleo de Scrum. Cada Sprint es un pequeño proyecto con el marco de tiempo necesario para obtener un producto funcional, con una duración máxima de un mes. Un Sprint tiene un objetivo a desarrollar y un plan flexible para lograrlo, y, finalmente, un producto resultante del mismo.

- **Planificación del Sprint:** Reunión al inicio de cada Sprint para planificar el trabajo que se realizará en el mismo. Todo el Equipo Scrum participa en esta reunión. En ella se deciden los componentes del Product Backlog que se llevarán a cabo en el Sprint y se diseña un plan para la realización de los mismos. Estos componentes podrán ser divididos en el caso de que sean demasiado complejos para realizarse en un Sprint.

- **Scrum diario:** Es un evento de corta duración (15 minutos) realizado por el Equipo de Desarrollo en el que se planean las actividades para las próximas 24 horas. Esta reunión facilita la organización interna del equipo y la visión del progreso hacia la meta del Sprint.

- **Inspección del Sprint:** Con el final de un Sprint se llevan a cabo dos reuniones, la Inspección del Sprint y la Retrospectiva del Sprint. En la Inspección del Sprint se presenta el producto incremental a los clientes y a todo el Equipo Scrum. Basándose en este producto y en los cambios al Product Backlog durante el Sprint, los asistentes colaboran en los siguientes objetivos a alcanzar para optimizar el valor del proyecto.

- **Retrospectiva del Sprint:** En esta reunión, ubicada entre la Inspección del Sprint presente y la próxima Planificación de Sprint, el Equipo Scrum revisa el desarrollo del Sprint realizado, para realizar una mejora continua del proceso.

2.1.4 Scrum en conjunto

Dados los distintos eventos y documentos que componen Scrum, en la figura 2.1 puede verse un diagrama que muestra su implementación. Esta implementación consta de los siguientes pasos:

1. El Product Owner, tras una serie de reuniones con los clientes, escribe el Product Backlog del proyecto, conteniendo los requisitos de la aplicación.

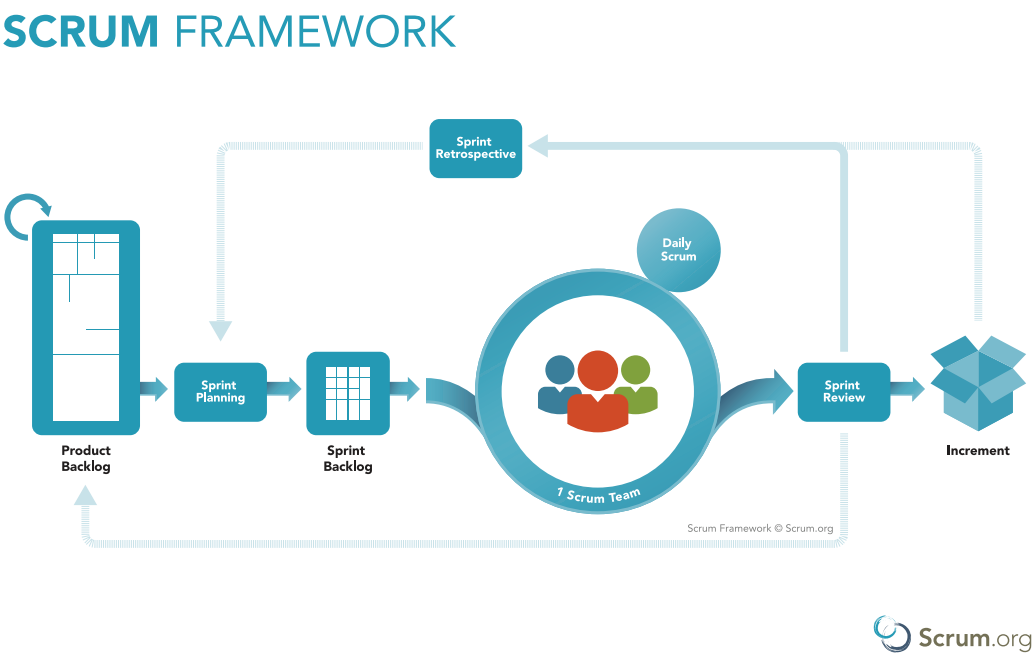

Figura 2.1: Vista gráfica de la implementación de Scrum a nivel de equipo [2]

2. Se realiza la planificación del primer Sprint y se obtiene un Sprint Backlog en la reunión de Planificación del Sprint.

3. El Equipo de Desarrollo ejecuta el Sprint a lo largo de un período de tiempo, manteniendo reuniones diarias.

4. Al finalizar el Sprint se llevan a cabo dos reuniones: la Inspección del Sprint y la Retrospectiva del Sprint.

5. Se repite el proceso de Planificación de Sprint, realización del mismo y reuniones hasta que el Product Backlog sea completado.

2.2 Cambios en la metodología Scrum

Debido a la naturaleza de este proyecto, siendo un Trabajo de Fin de Grado, se llevaron a cabo ciertos cambios a la metodología Scrum para adaptarla a esta situación.

2.2.1 Roles en Scrum

El Equipo Scrum se compone de los directores del Trabajo de Fin de Grado y un alumno. Los directores llevaron a cabo los roles de cliente, Product Owner y Scrum Master, mediante

la definición de los requisitos del proyecto así como la asistencia al Equipo de Desarrollo. El alumno fue el único integrante del equipo de desarrollo.

2.2.2 Reuniones en Scrum

Debido a que este proyecto se trata de un trabajo de fin de grado, y a la inexperiencia del alumno en proyectos de esta escala, se ha modificado el sistema de reuniones de Scrum. Para mantener cierta regularidad y un ambiente estructurado, se han llevado a cabo reuniones semanales. En caso de haber acabado un Sprint, se realizaron seguidas las reuniones de Inspección del Sprint, Retrospectiva del Sprint y Planificación de Sprint. En caso contrario, se llevó a cabo una reunión de seguimiento y resolución de dudas. Además, se usó el contacto mediante correo electrónico y reuniones virtuales en caso de imposibilidad de una reunión física.

2.3 Planificación y seguimiento

2.3.1 Planificación del proyecto

El proyecto se dividió en cinco Sprints: análisis del estado del arte en RS, análisis y preprocesado de la base de datos, representación de lenguaje de los comentarios, predicción de comentarios y escritura de la memoria. Además, antes de comenzar con el proyecto ha sido necesaria la planificación del mismo y la escritura del anteproyecto. En la figura 2.2 se puede observar el diagrama de Gantt que contiene la planificación de este proyecto. Para su mejor visualización, se incluye una copia del diagrama en mayor tamaño en los apéndices, la figura A.1. A continuación se explica brevemente cada Sprint y sus requisitos asociados.

- **Sprint 1 - Análisis del estado del arte en RS:** en este Sprint se analizaron los diferentes papers publicados sobre RS que utilicen texto como núcleo del análisis, con el objetivo de conseguir una idea general del estado del arte en este campo, así como de encontrar posibles mejoras o alternativas a lo planteado en este proyecto. Además, se realiza un pequeño análisis del funcionamiento de los grandes RS utilizados en la actualidad por Youtube y Amazon.

- **Sprint 2 - Análisis y preprocesado de la base de datos:** el objetivo de este Sprint es conseguir una base de datos y procesarla a un formato que se pueda utilizar con el algoritmo Doc2Vec. Para ello será necesario que analizar la literatura sobre Doc2Vec para conocer qué requisitos debe tener esta base de datos, obtener una base de datos, procesarla y analizar que cumple dichos requisitos.

- **Sprint 3 - Representación densa del lenguaje de los comentarios:** el objetivo de este Sprint es mapear los comentarios textuales que conforman la base de datos a una representación vectorial de baja dimensión utilizando Doc2Vec. Para ello es necesario analizar el funcionamiento de Doc2Vec, implementar el algoritmo, estudiar e implementar un método para evaluar su rendimiento y experimentar con los parámetros de este algoritmo para lograr el mejor rendimiento posible.

- **Sprint 4 - Predicción de comentarios:** el objetivo de este Sprint es, una vez hallada la representación vectorial de los comentarios, predecir los vectores asociados a comentarios de usuarios a hoteles donde no hayan estado, y posteriormente transformar estos vectores de nuevo a comentarios en texto. Para esto es necesario analizar dichos vectores, analizar la literatura al respecto, estudiar e implementar diversos algoritmos para la predicción, estudiar e implementar un método para evaluar su rendimiento y experimentar con los parámetros de este algoritmo para lograr el mejor rendimiento posible.

- **Sprint 5 - Escritura de la memoria:** en este Sprint se escribe la memoria del proyecto.

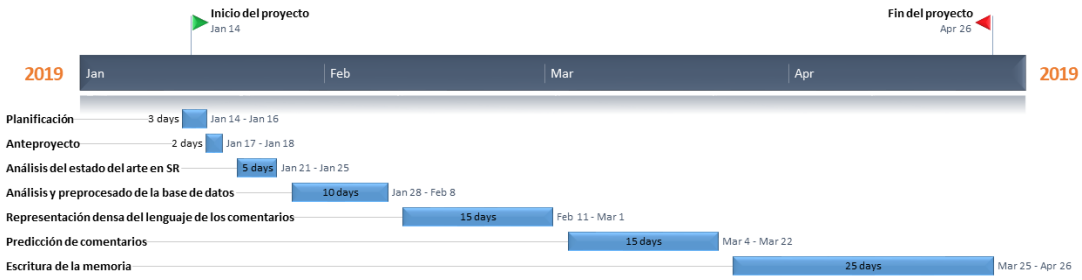

Figura 2.2: Planificación inicial del proyecto

2.3.2 Seguimiento del proyecto

Este proyecto ha sufrido una serie de retrasos, el principal debido a una larga baja por razones médicas, que hizo que el Sprint 3, se dejase inacabado. Siguiendo las reglas de Scrum, se hicieron las reuniones pertinentes y se realizó un nuevo Sprint con los requisitos incompletos del anterior. Una vez acabado este Sprint, debido a que se podía hacer una mejora de los resultados del algoritmo utilizado (Doc2Vec) mediante la repetición del algoritmo con distintos parámetros, se añadió un requisito extra al siguiente Sprint, que sería la ejecución y análisis del experimento con dicha mejora, y se amplió el tiempo inicialmente planificado para este

Sprint. Puede verse una representación gráfica del seguimiento en la figura 2.3. Para su mejor visualización, se incluye una copia del diagrama en mayor tamaño en los apéndices, la figura A.2.

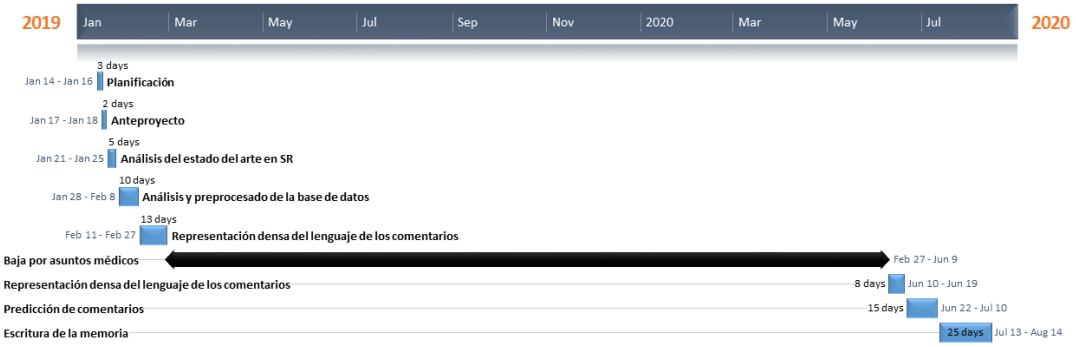

Figura 2.3: Seguimiento del proyecto

2.4 Estimación de costes del proyecto

En esta sección se desglosarán los costes estimados del desarrollo del proyecto. En este proyecto existen dos actores: el director y el alumno. Hay por lo tanto dos perfiles de trabajo: jefe de proyecto e ingeniero informático/científico de datos. Se ha estimado su salario en 35€/hora y 15€/hora, respectivamente. Para el cálculo del tiempo trabajado en los Sprints, se ha tenido en cuenta una jornada laboral de 5 horas. En la tabla 2.1 puede verse el desglose de horas de trabajo y costes asociados a cada Sprint.

El trabajo del jefe de proyecto se estimó en unas 50 horas en total, teniendo en la tabla 2.2 un desglose del salario por perfil y el coste total asociado al proyecto.

Sprint	Horas de trabajo	Coste en euros
Sprint 1 - Análisis del estado del arte en RS [20]	25	375
Sprint 2 - Análisis y preprocesado de la base de datos	50	750
Sprint 3 - Representación densa del lenguaje de los comentarios	65	975
Sprint 4 - Representación densa del lenguaje de los comentarios 2	40	600
Sprint 5 - Predicción de comentarios + espera experimento	110	1650
Sprint 6 - Escritura de la memoria	150	2250
Total de todos los Sprints	440	6600

Tabla 2.1: Desglose del horas y salario del ingeniero informático/científico de datos por Sprint

Perfil del trabajador	Coste total en euros
Ingeniero informático/científico de datos	6600
Jefe de proyecto	1750
Total	8350

Tabla 2.2: Costes del proyecto

Capítulo 3

Estado del arte

En este capítulo se analiza el estado del arte de los RS en relación al uso de texto como elemento nuclear del análisis. Además, se realiza un pequeño análisis del funcionamiento de los grandes RS utilizados en la actualidad por Youtube y Amazon.

3.1 Estado del arte en RS con análisis de texto como elemento nuclear

Algunos de los primeros RS que hacían uso de técnicas de Procesamiento de Lenguaje Natural (NLP por sus siglas en inglés) tenían propósitos diferentes de aprovechar reseñas de usuarios.

Chai et al. [21] presentaba un sistema de diálogo interactivo que encuentra un ordenador portátil adecuado a las especificaciones provistas por el usuario en lenguaje natural. Por lo que, técnicamente hablando, es una interfaz a una base de datos en lugar de un RS. Por otro lado, el sistema de Fleischman y Hovy [22] recomienda películas sin ninguna información previa de las preferencias del usuario. Para hacerlo, obtiene películas similares a una dada comparando los resúmenes de su trama usando un modelo Bag-of-words simple. Este modelo consiste en la simplificación de textos en lenguaje natural a un multiset de palabras, obviando la relación gramatical de las mismas. Sin embargo, este sistema es un RS puramente basado en contenido, en el que no se utilizan reseñas generadas por usuarios.

Leung et al. [14] fueron de los primeros en notar la utilidad de reseñas en texto para complementar las valoraciones numéricas de los RS. En particular, se centran en dominios donde no hay o hay pocas valoraciones numéricas, pero sí hay acceso a reseñas en texto de las que se pueden inferir dichas valoraciones. Emplea un enfoque usando análisis de sentimiento (clasificación masiva de documentos de manera automática, en función de la connotación positiva o negativa del lenguaje ocupado en el documento, basada en métodos estadísticos) para con-

densar las reseñas en valoraciones numéricas, que son usadas para filtrado colaborativo en un RS convencional.

El trabajo de Jakob et al. [23] fue pionero en una línea de trabajo centrada en usar la información de las reseñas textuales, no para condensarlas en una valoración numérica, sino para mejorar la predicción de la valoración mediante la extracción de palabras relevantes del texto que muestran similitudes entre productos o usuarios, usando clustering de Latent Dirichlet Allocation (LDA), un modelo generativo que permite explicar conjuntos de observaciones mediante grupos no observados que explican por qué algunas partes de los datos son similares (en este caso, la similitud son las palabras relevantes extraídas). Las aproximaciones de McAuley y Leskovec [24], Ling et al. [25] y Bao et al. [26] mantienen una línea similar: todas presentan diversas formas de combinar datos de valoraciones numéricas con análisis de sentimientos (modelos estadísticos que extraen los "temas" abstractos que ocurren en un conjunto de documentos, por ejemplo: cocina, salud, deporte, etc.) entrenados en textos de reseñas con el propósito de aumentar la precisión de la predicción de las valoraciones numéricas. Merece la pena recalcar que estas aproximaciones no intentan generar texto como salida, y el procesado de las reseñas es, de nuevo, limitado a un modelo Bag-of-words, es decir, explotan las similaridades entre reseñas basándose únicamente en su contenido, sin tener en cuenta el orden de las palabras o la estructura lingüística.

Wang et al. [15] comparten el mismo procesado Bag-of-words para mejorar la predicción de valoraciones numéricas, pero utilizan un modelo bayesiano jerárquico implementado con técnicas de deep learning en lugar de topic models. Este modelo es el llamado Colaborative Deep Learning (CDL), que une el filtrado colaborativo de las puntuaciones numéricas de usuarios sobre productos con aprendizaje de características profundo (un conjunto de técnicas que permite que un sistema descubra automáticamente las representaciones necesarias para la detección o clasificación de características a partir de datos sin procesar, mediante el uso de redes neuronales artificiales) del contenido textual. Entre las aproximaciones de este tipo, la más similar a la de este trabajo procede de Gaillard et al. [17]. Aunque sigue utilizando el sistema Bag-of-words para predecir recomendaciones, esta aproximación devuelve una "argumentación" del porqué de la recomendación. Dicha argumentación toma la forma de una serie de palabras o expresiones que tienden a aparecer en las reseñas, en lugar de una reseña coherente, y no se intenta evaluar la precisión de la misma.

Más recientemente se han presentado diversas aproximaciones que dejan a un lado la representación de los textos mediante el modelo Bag-of-words, utilizando word embeddings (representación densa de palabras a vectores, explicado en detalle en la sección 5.3.1) en su lu-

gar. Esto último lo intentaron Ni et al. [18], siendo esta una de las aproximaciones en literatura más cercana al objetivo de este trabajo. Los autores implementan un RS que utiliza reseñas de usuarios como el núcleo de la predicción. Estas predicciones utilizan valoraciones numéricas y texto de reseña. La predicción de las valoraciones se lleva a cabo mediante una factorización numérica y una representación simple de los textos. Los autores utilizan Word2vec [27] (explicado en la sección 5.3.1) para representar las palabras de las reseñas de usuarios como Word-vectors. Después, computan el valor medio de estos vectores para representar toda la reseña. Los autores utilizan un Long Short-Term Memory (LSTM) para generar los textos predichos a nivel de caracter. Un LSTM es un tipo de Red Neuronal Recurrente (RNN por sus siglas en inglés, red con bucles de retroalimentación que permiten que la información persista durante un número de iteraciones de entrenamiento), capaz de recordar información a más largo plazo que las RNN tradicionales. Esta aproximación sin embargo dificulta que los textos predichos tengan coherencia.

Las diferencias principales entre la aproximación de este trabajo y el paper de Ni et al. son las siguientes:

- La utilización del algoritmo Doc2vec [28] (explicado en detalle en la sección 5.3.2) para predecir la representación vectorial de las reseñas textuales, un algoritmo creado específicamente para condensar la información de documentos enteros en lugar de la aproximación de la media de los vectores de Word2vec.
- La predicción de los textos utiliza una regresión que garantiza que el texto generado sea coherente, ya que el texto que se predice es uno de los vistos en entrenamiento.

Otra aproximación similar a la de este proyecto es Deep Cooperative Neural Networks (DeepCoNN) [16]. Zheng et al. crearon este algoritmo capaz de predecir la valoración de un usuario a un producto utilizando únicamente textos de reseñas de usuarios. Para ello, utilizan dos redes neuronales para perfilar usuarios y productos, y las concatenan en una última capa. Para codificar los textos, utilizan una matriz de word embeddings, con un vector para cada palabra de la reseña. Dichos vectores no se extraen de las reseñas, si no que son vectores preentrenados con otros textos. Chatherine y Cohen [29] muestran que el rendimiento de DeepCoNN es muy dependiente de tener una reseña del usuario y producto para predecir un valor numérico, lo cual es una suposición poco realista. Ellos extendieron el modelo con una capa extra que transforma las representaciones de cada par usuario y producto a una representación del par. El objetivo de estos modelos sigue siendo la predicción de valoraciones numéricas, no crear un texto.

Chong Chen et al. proponen el algoritmo NARRE (Neural Attentional Regression model

with Review-level Explanations) [30] basándose en DeepCoNN. El objetivo de NARRE no es sólo obtener valoraciones de usuarios a productos para predecir sus intereses; también presenta al usuario con la reseña más importante para predecir dicha valoración. Es decir, de forma similar a este proyecto, presenta una reseña en lenguaje natural para explicar la recomendación, ahorrando al usuario la búsqueda entre cientos de reseñas para informarse de los aspectos más relevantes del producto que se le recomienda. Similar a DeepCoNN, NARRE implementa dos redes neuronales profundas para perfilar usuarios y productos, pero tiene diferencias importantes. También utiliza word embeddings preentrenados, pero convierte cada reseña en un único vector aplicando una red neural convolucional (CNN por sus siglas en inglés, una red neuronal muy utilizada en análisis de imagen) a las matrices de word embeddings, y utiliza dichos vectores para perfilar productos y usuarios. Con esto se consigue que un factor importante en una reseña no domine sobre el resto de características. Además, se implementa una capa de atención en la red para descartar aquellas reseñas que no aporten información relevante durante el entrenamiento.

Las diferencias principales entre la aproximación de este trabajo y NARRE son las siguientes:

- La utilización del algoritmo Doc2vec [28] para predecir la representación vectorial de las reseñas textuales, un algoritmo creado específicamente para condensar la información de documentos enteros en lugar de la aproximación de una matriz de word embeddings para cada palabra de la reseña que después se convoluciona en un vector con una CNN.
- El perfilado de usuarios y productos se entrena a la vez que Doc2vec, separando la predicción en un paso diferente.
- En lugar de usar embeddings preentrenados, en este proyecto se extrae toda la información directamente de la base de datos utilizada, logrando mayor especificidad.
- NARRE es capaz de descartar reseñas poco relevantes para el perfilado de usuarios y productos, lo cual podría ser un buen aporte a futuras iteraciones del proyecto.

3.1.1 Estado del arte posterior al inicio del proyecto

Debido al largo intervalo de tiempo entre el inicio de este proyecto y su finalización, y al ubicarse en un campo de estudio en auge, han surgido diversos papers relevantes que no se tuvieron en cuenta durante la realización del mismo. En concreto, en el ámbito de las explicaciones textuales personalizadas en los RS, han surgido varias alternativas con mejor rendimiento que NARRE [30].

Cong et al. [31] propone HANN (Hierarchical Attention-Based Network) como alternativa a NARRE. HANN es capaz de generar explicaciones considerando la contribución de las reseñas a nivel palabra, además de a nivel de reseña como NARRE. También se utilizan dos redes neuronales, en este caso Gated Recurrent Units (GRU, un subtipo de LSTM), para perfilar usuarios y productos, cada una de ellas con dos capas de atención, una a nivel de palabra dentro de cada reseña y otra a nivel de reseña. Este algoritmo es capaz de distinguir no sólo qué reseñas son más relevantes, si no también las palabras más relevantes en cada reseña (ej: "barato", "estupendo").

Xu Chen et al. [32] crearon otro RS capaz de explicar en lenguaje natural el por qué de las recomendaciones, el llamado Dynamic Explainable Recommender (DER), que añade temporalidad respecto a NARRE. Se modelan las preferencias de los usuarios de forma dinámica en el tiempo mediante el uso de una GRU, y se construye el perfil de los productos del RS analizando el texto de sus reseñas mediante una CNN. De esta forma es capaz de aprender la información relevante de las reseñas según las preferencias del usuario en un punto del tiempo y es capaz de dar explicaciones en lenguaje natural sobre dichas preferencias. Estas explicaciones son frases seleccionadas de diferentes reseñas, por lo que padecen un problema similar al paper de Ni et al. [18], y es que pueden presentar una falta de coherencia al ser analizadas en conjunto. Este paper supone un caso de estudio relevante para futuras iteraciones del proyecto, al ser capaz de explotar información temporal para ajustarse a las preferencias dinámicas de los usuarios, mientras que en el presente proyecto no se tiene en cuenta dicha información.

3.2 Uso de RS en el mercado

Como se explicó en la introducción al proyecto (ver sección 1), los RS son explotados con gran éxito fuera del ámbito académico por grandes empresas multinacionales. Conviene por tanto explorar cómo funcionan los RS de estas empresas, que están en constante desarrollo y que han demostrado su eficacia en el mercado. En este proyecto se analizaron los RS de dos empresas, Youtube y Amazon, ya que han publicado papers donde explican el funcionamiento de sus algoritmos, aunque no en detalle.

Amazon es una compañía multinacional centrada en el comercio electrónico, computación en la nube, streaming e inteligencia articial. Sin embargo, la parte relevante para este proyecto es la de comercio electrónico. En ella, Amazon tiene dos tipos de usuarios, los vendedores y los compradores. Los compradores pueden buscar productos, ya sea por palabras clave, categorías o características concretas (aspiradoras con o sin cable, rango de precio etc.), añadir

los productos a una lista de deseos, comprar productos y opinar sobre ellos, mediante una valoración numérica, un comentario y opcionalmente una serie de fotografías (puede verse un ejemplo en la figura 1.2).

La finalidad del RS de la web de comercio electrónico de Amazon es recomendar productos a sus usuarios para que estos los compren. Para esto, hacen uso de un RS de filtrado colaborativo "item a item"[33] [34], donde se deriva una lista de productos similares a los productos buscados, vistos, comprados y calificados por un usuario para posteriormente combinarlos en una lista de recomendaciones y mostrar al usuario los más populares o correlacionados. Sin embargo, Amazon no detalla con precisión qué tipo de datos utiliza para encontrar productos similares, ni qué considera similitud entre dos productos, aunque se da un ejemplo. En este ejemplo, se utilizan como características de cada producto un vector de M dimensiones que se corresponde con los usuarios que han comprado el producto, y se mide la similitud entre ellos mediante la similitud coseno (medida de similitud entre vectores utilizada en este proyecto y explicada en la sección 5.3.4).

El punto clave de este tipo de filtrado es la escalabilidad y mejora de rendimiento que produce el crear las tablas de similitud entre productos offline, mientras la parte online del algoritmo (encontrar productos similares a aquellos con los que el usuario ha interaccionado) escala independientemente del número de usuarios y productos, sólo es dependiente de los productos con los que el usuario interacciona.

Por su parte, Youtube es una página web enfocada al contenido audiovisual, que permite a sus usarios subir, ver, puntuar, compartir, crear listas de reproducción, reportar, comentar en vídeos y suscribirse a otros usuarios. Las puntuaciones son simples, "me gusta" y "no me gusta". Los comentarios a los vídeos constan de un texto, se pueden puntuar y permiten respuestas de otros usuarios. El objetivo del RS de Youtube es recomendar a sus usuarios nuevos vídeos para ver y maximizar el tiempo que un usuario ve vídeos.

El equipo de Youtube ha descrito cómo funciona su algoritmo de recomendación en varias ocasiones, siendo la última en 2016 [35], y aunque el algoritmo ha cambiado desde entonces, el equipo de Youtube ha confirmado que la base sigue siendo la misma [36]. Aunque esta no es una explicación completa y detallada, sí deja ver el funcionamiento general del RS. Al igual que en el caso de Amazon, se trata de un RS de filtrado colaborativo "item a item". Este RS hace uso esencialmente de dos redes neuronales: una de generación de candidatos y otra de clasificación. La red de generación de candidatos tiene como objetivo reducir los vídeos de Youtube a unos cientos que sean relevantes para el usuario (acabando con el problema de escala que supone el enorme tamaño de la videoteca de Youtube), y a continuación la red de clasificación los clasifica en función del tiempo esperado de visualización que tendrá el usuario en el vídeo. Existen cientos de parámetros que utilizan para alimentar las redes, siendo

los más importantes aquellos que describen la interacción previa de un usuario con el vídeo y otros similares (cuántos vídeos de ese canal ha visto el usuario recientemente, popularidad del vídeo etc.).

Cabe destacar que tanto Amazon como Youtube resuelven el problema de temporalidad (que los productos/vídeos más populares del pasado continúen recomendándose para siempre) al utilizar la edad de una instancia de entrenamiento como parámetro en la red. Esta característica no se utilizó en este proyecto, pero sí se recomienda para futuras iteraciones.

La similitud más destacable de los RS de Amazon y Youtube con este proyecto es el uso de representaciones densas o embeddings (explicados en la sección 5.3) para representar las características no representables directamente de forma numérica, incluyendo los textos (por ejemplo, título y descripción de los vídeos en Youtube).

Sin embargo, no existe referencia al uso de los comentarios de usuarios en los productos de Amazon o vídeos de Youtube como parámetro a utilizar al entrenar las redes, más allá de su uso para determinar popularidad (velocidad y cantidad en que un vídeo genera comentarios, likes y reproducciones). Por lo tanto, aunque no se puede descartar su uso, sí puede asumirse que no es un elemento central de la predicción, y es precisamente en la explotación de dicha información en lo que se basa este proyecto.

Capítulo 4

Tecnologías

En este capítulo introducen brevemente las diversas tecnologías empleadas a lo largo del desarrollo del proyecto. En primer lugar se expone el lenguaje de programación utilizado junto con algunas bibliotecas externas básicas usadas a lo largo de todo el proyecto. Posteriormente, siguiendo el esquema del proyecto (ver figura 1.4), se exponen las bibliotecas usadas para llevar a cabo el procesamiento de la base de datos, la implementación de Doc2vec y las bibliotecas utilizadas durante la predicción de reseñas. También se comenta la biblioteca utilizada para optimización de hiperparámetros (explicado en la sección 5.4) que se aplicó al algoritmo Doc2vec. A continuación se describen brevemente los entornos de desarrollo utilizados para la creación del proyecto y la memoria. Por último, se comenta la plataforma utilizada para la ejecución del proyecto en el Centro de Supercomputación de Galicia (CESGA).

4.1 Lenguaje de programación y bibliotecas generales

En esta sección se introduce de manera breve el lenguaje Python, así como una serie de bibliotecas utilizadas a lo largo de todo el proyecto.

Python

Python [37] es un lenguaje de programación interpretado multiparadigma de alto nivel para propósito general. Soporta programación imperativa, orientada a objetos y, en menor medida, funcional. Tiene un sistema de tipado dinámico, asigna memoria cuando es necesaria y utiliza un recolector de basura que revisa la memoria desreferenciada para liberarla. Una comunidad global de desarrolladores sin ánimo de lucro, la Python Software Foundation, mantiene la implementación de referencia, CPython, de código libre. Es un lenguaje ideal para una metodología de desarrollo ágil (como es Scrum, la metodología usada en este proyecto), debido a su sencillez, facilidad de lectura y su extensa colección de bibliotecas, que permiten desde crear interfaces grá-

ficas a computación científica.

Numpy

NumPy [38] es una biblioteca de código abierto para Python, que añade soporte para uso de arrays y matrices de grandes dimensiones, así como una colección de funciones de alto nivel para operar con ellas.

More Itertools

More Itertools [39] es una biblioteca de código y colaboración abierta con múltiples funciones de alto nivel para trabajar con los iterables de python.

Matplotlib

Matplotlib [40] es una paquete de gráficos 2D usado para el desarrollo de aplicaciones, scripting interactivo y generación de imágines de calidad al nivel de publicaciones académicas en Python.

Dill

Dill [41] es un paquete que extiende la funcionalidad de Pickle, un módulo para la serialización y des-serialización de objetos en Python. Aunque en este proyecto se utiliza para guardar dichos objetos en archivos, se desarrolló con la intención de enviar objetos de Python a través de la red.

4.2 Bibliotecas específicas del proyecto

En esta sección se introducen diversas bibliotecas utilizadas a lo largo del proyecto, y se indica su uso.

NLTK

El Natural Language Toolkit, más comúnmente llamado NLTK [42], es un compendio de bibliotecas y programas para procesado de lenguaje natural (NLP) simbólico y estadístico para el idioma Inglés escrito en Python. Se utilizó en el proyecto para el preprocesado del texto de las reseñas de la base de datos.

Gensim

Gensim [43] es una biblioteca código abierto para el NLP y topic modeling utilizando

técnicas modernas de aprendizaje automático estadístico. Está implementada en Python y Cython (un lenguaje compilado designado para dar un rendimiento parecido al código en C pero escrito mayormente en Python), con un énfasis en paralelismo y velocidad, para poder manipular grandes cantidades de datos de forma eficiente y escalable. Cabe destacar que esta biblioteca no utiliza GPU para su funcionamiento, aunque está en su hoja de ruta la implementación en GPU para aumentar la velocidad de sus algoritmos. En este proyecto se utilizó la implementación del algoritmo Doc2vec de esta biblioteca.

Keras

Keras [44] es una biblioteca de código abierto con una API de alto nivel para la creación de redes neuronales de forma simple y accesible. Esta biblioteca puede utilizar CPU o GPU sin requerir cambios en el código, y permite el entrenamiento distribuido en múltiples máquinas para aumentar la velocidad. Puede funcionar utilizando como backend Theano, PlaidMl, el Microsoft Cognitive Toolkit o Tensorflow, que fue el utilizado en este proyecto. Esta biblioteca contiene numeras implementaciones para los bloques de construcción más usados en las redes neuronales, como capas, objetivos, funciones de activación, optimizaciones y una gran variedad de herramientas para el trabajo con imágenes y texto. Se utilizó Keras para la implementación de la red neuronal de regresión que predice las reseñas en el proyecto.

Hyperopt

Hyperopt [45],[46] es una biblioteca de código abierto para la optimización de parámetros sobre espacios de búsqueda complicados, que pueden incluir parámetros discretos o numéricos en diversas distribuciones y dimensiones condicionales. A pesar de lo que su limitada documentación puede decir de esta biblioteca, tiene un potencial elevado debido a la simpleza de su uso y la capacidad de utilizar parámetros condicionales y anidados. Se utilizó la biblioteca Hyperopt para optimizar los hiperparámetros de entrada del algoritmo Doc2vec.

Visualización de hiperparámetros con hyperopt

Para la visualización de los parámetros escogidos por la biblioteca Hyperopt anteriormente citada, se utilizó y modificó parte de un repositorio github con licencia de software libre [47], originalmente creado como ejemplo para usar la biblioteca Hyperopt con Keras, Tensorflow y Tensorboard, y visualizar los resultados.

4.3 Entornos de desarrollo

En el desempeño de este proyecto se han utilizado diversas herramientas para facilitar su implementación y diseño, expuestas brevemente a continuación.

Pycharm

Pycharm [48] es un entorno de desarrollo integrado (IDE) multiplataforma utilizado para la programación en Python. Tiene una versión comercial y una versión de software libre llamada Pycharm Community, más limitada, que fue la usada en este proyecto. Este entorno proporciona, entre otras cosas, análisis de código y un depurador gráfico. Puede verse un ejemplo de esta interfaz en la figura 4.1.

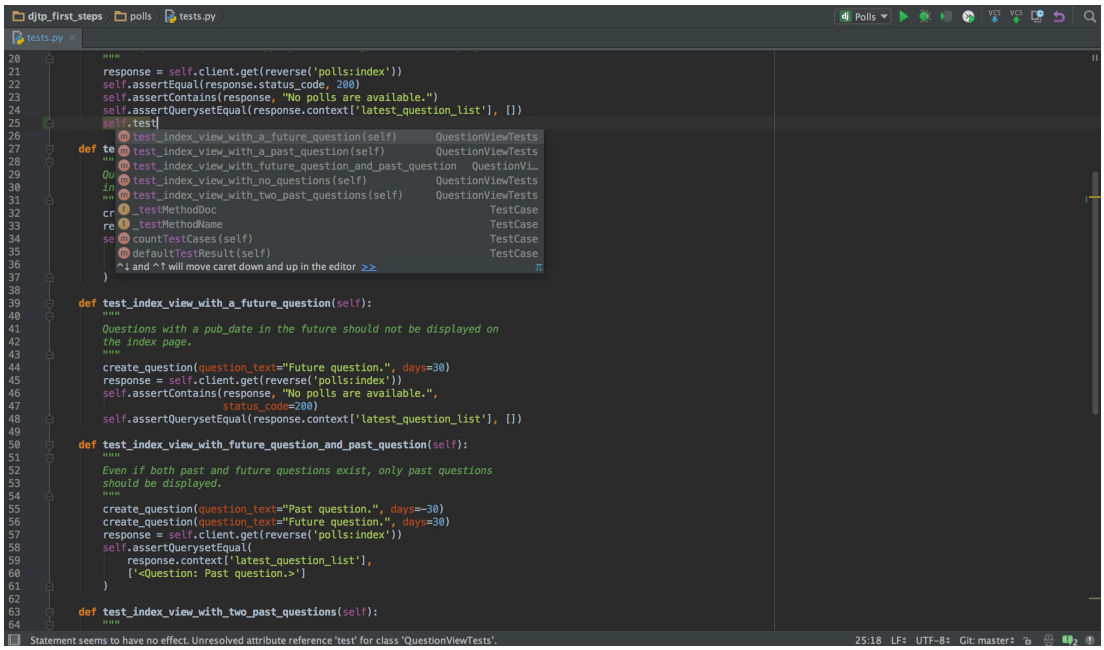

Figura 4.1: Interfaz de Pycharm

Draw.io

Draw.io [49] es un compendio de tecnologías open source para la creación de diagramas online, con integración para guardar proyectos en diversos almacenamientos en la nube. Cuenta también con una versión offline, y la versión online permite utilizar el almacenamiento interno del dispositivo que la usa para almacenar los diagramas. Esta aplicación es simple pero extremadamente versátil y fácil de utilizar. Puede verse una captura de la aplicación online de draw.io en la figura 4.2.

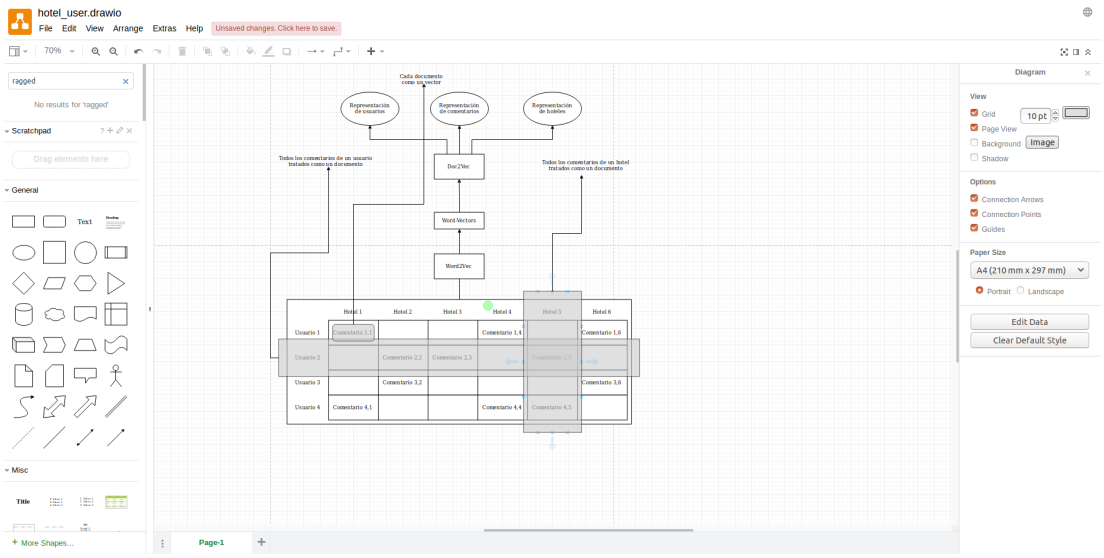

Figura 4.2: Interfaz de draw.io

Office Timeline

Office Timeline [50] fue creado como un complemento para Microsoft PowerPoint para crear líneas de tiempo y diagramas de Gantt simples y altamente personalizables. También presenta una versión online, de cuyo uso puede verse un ejemplo en la figura 4.3.

4.4 Ejecución del proyecto en el Centro de Supercomputación de Galicia (CESGA)

Debido a la cantidad ingente de datos necesarios para utilizar el algoritmo Doc2vec y el largo tiempo para su procesamiento, para la ejecución de este proyecto se utilizaron nodos del servidor de cálculo “Finis Terrae II” del CESGA, financiado por el Ministerio de Economía y Competitividad, la Xunta de Galicia y el FEDER (Fondo Europeo de Desarrollo Regional). Para el acceso a los mismos se envió una solicitud a las instalaciones con la aprobación del director del departamento de Computación de la Universidad de la Coruña.

El acceso a los servidores se realiza mediante un cliente SSH que permite encriptar la información que se envía y recibe del mismo, además de ejecutar comandos de forma remota y enviar ficheros. El servidor utiliza el sistema SLURM (Simple Linux

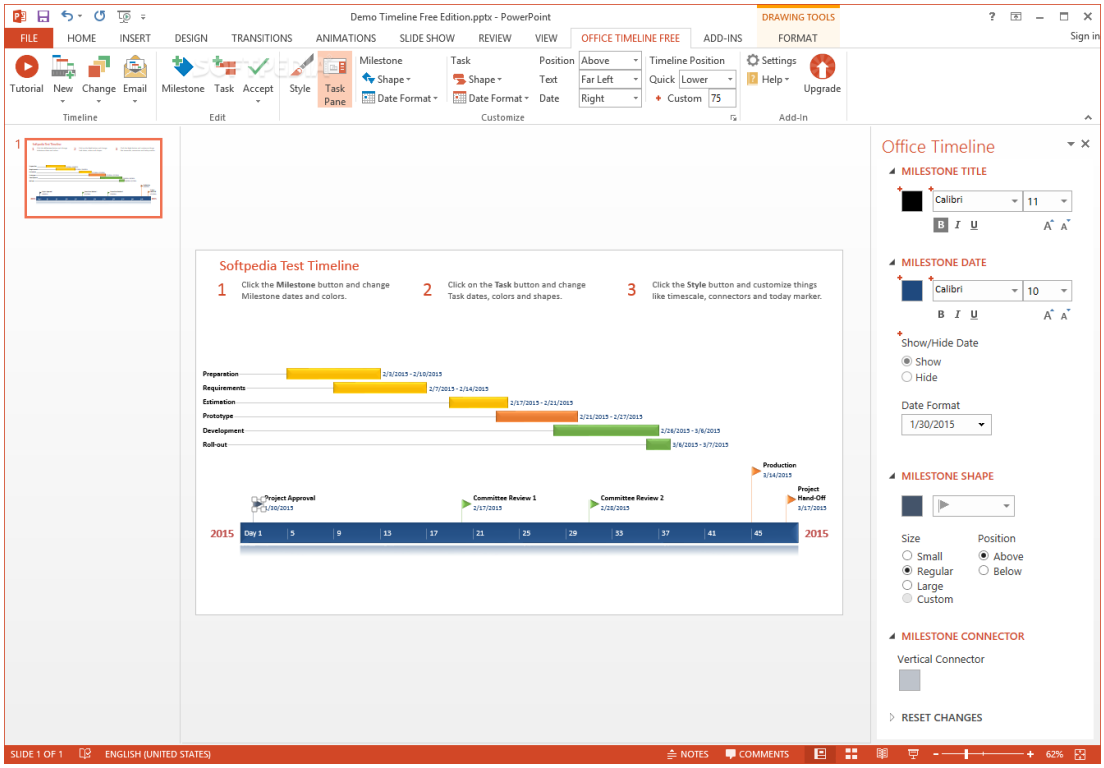

Figura 4.3: Interfaz de Office Timeline

Utility for Resources Management) [51] para gestionar tareas de los usuarios en los distintos nodos y clústeres, que proporciona un entorno para asignar recursos de forma exclusiva o compartida a las tareas mediante una cola de tareas pendientes con distintas prioridades y limitaciones.

Varios comandos útiles de SLURM son:

sbatch : utilizado para enviar un script que contiene una tarea al sistema de colas.

squeue : utilizado para ver el estado actual de tus tareas.

sinfo : utilizado para ver información de los distintos nodos y particiones del sistema.

tsacctmgr : utilizado para ver (y en caso de ser administrador modificar) la información de la cuenta de un usuario, incluyendo los nodos a los que tiene acceso y la prioridad de sus tareas.

module : utilizado para acceder de forma simple a cualquier pieza de software instalada por un administrador del sistema (por ejemplo, cargar CUDA y CUDNN para la aceleración por GPU).

Un ejemplo de script para utilizar el sistema SLURM:

```
#SBATCH --nodes=1 # Un nodo
#SBATCH -ntaks=1 # Una tarea
#SBATCH --cpus-per-task=4 # Cuatro núcleos de CPU por tarea
#SBATCH --gpus-per-task=4 # Cuatro GPUS por tarea
#SBATCH --partition=cola-corta # Partición de los nodos a la que enviar las tareas
#SBATCH --qos=interactive # Prioridad de las tareas
#SBATCH --time=0-10:00:00  # Tiempo de  ejecución
#SBATCH --mail-type=begin # Envía un correo cuando el trabajo inicia
#SBATCH --mail-type=end # Envía un correo cuando el trabajo finaliza
#SBATCH --mail-type=fail # Envía un correo cuando el trabajo falla
#SBATCH --mail-user=mi_nombre@udc.es # Dirección a la que se envía
#SBATCH --output=/path_to_home/testing.out # Fichero al que redirigir la salida standard
#SBATCH --error=/path_to_home/testing.err # Fichero al que redirigir mensajes de error
module load CUDA # Software que precargar en el entorno de las tareas
module load CUDNN
source /path_to_home/virtual_env/bin/activate # Activación del entorno virtual en el que
 se ejecutará el código

srun /path_to_home/virtual_env/bin/python3.6 /path_to_home/code/main.py
# Tarea a ejecutar
```

Figura 4.4: Ejemplo de script para enviar con el comando sbatch

Capítulo 5

Materiales y métodos

En este capítulo se expone y analiza el único material base utilizado durante el proyecto, una base de datos de la web TripAdvisor, así como los diversos métodos estudiados para su aplicación en el proyecto.

Conviene recordar el esquema del proyecto (ver figura 1.4): en primer lugar se obtiene la base de datos y se preprocesa, en segundo lugar se obtiene una representación densa del lenguaje de las reseñas de la base de datos mediante el uso del algoritmo Doc2vec, y por último, se aplica un algoritmo de aprendizaje profundo para predecir las reseñas de usuarios a hoteles que no han visitado, y se miden los resultados.

5.1 Base de datos

Para este proyecto se empleó la base de datos (DB por sus siglas en inglés) de comentarios de TripAdvisor que la Universidad de Illinois tiene disponible públicamente para analizar [52]. Ocupa 2,1 GB y contiene ∼13.000 ficheros en formato JSON, del que se puede ver un ejemplo en la figura 5.1. Este formato, derivado de JavaScript, se utiliza para almacenar y transmitir objetos en forma de pares de nombre y valor, arrays y cualquier otro valor serializable. Una ventaja de este formato para el procesamiento de texto es que su codificación es legible de forma sencilla para los usuarios, por lo que se pueden abrir los archivos de forma directa en cualquier editor de texto para poder leerlos.

En concreto, cada fichero JSON de esta base de datos contiene datos sobre un hotel y una lista de usuarios con sus reseñas. La información de hotel incluye: nombre, identificador numérico, precio, localización, una URL con una foto del hotel y otra URL a la página del mismo en TripAdvisor. Sin embargo, un análisis exhaustivo muestra que muchos de los ficheros omiten parte de esta información, como la localización y el nombre, además de encontrar varios identificadores únicos para el mismo ho-

```
{
    "Reviews": {
        "0":{
            "Author": "Jeremy_Smith"
            "Content": "We enjoyed the hotel. The
                        rooms were clean."
            "Date": "March 29, 2012"
            "Review_id": "UR126946257"
        }
        "1":{
            "Author": "John Doe"
            "Content": "This hotel was horrible!"
            "Date": "July 3, 2010"
            "Review_id": "UR126946288"
        }
    }
    "HotelInfo": {
        "Name": "Pioneer Square Hotel"
        "Hotel_id": "72572"

    }
}
```

Figura 5.1: Ejemplo de formato JSON.

tel. Otro problema es la inconsistencia de los datos, con ciertos ficheros incluyendo URLs completas a TripAdvisor y otros tan sólo parciales, omitiendo la parte común: "http://www.tripadvisor.com".

Los datos de cada reseña de usuario contienen: el nombre de usuario en TripAdvisor, la fecha de la reseña, el título, el texto de la misma, la localización del usuario, un identificador para cada reseña y una serie de valoraciones numéricas de diferentes parámetros del hotel así como una valoración general. De nuevo, existe una enorme heterogeneidad de datos que no aparecen en ciertos hoteles o reseñas, así como la codificación de las mismas. Así, por ejemplo en las valoraciones numéricas, la cantidad de aspectos a evaluar varía de reseña a reseña, así como su codificación, ya que ciertas reseñas omiten los campos que de los que no tienen datos y otras asignan un valor negativo a los mismos. Otro aspecto inconsistente es la longitud de las reseñas en texto, con algunas reseñas de tan sólo una palabra y otras mucho más minuciosas y extensas. Un aspecto que podría ser interesante para su análisis en este proyecto podría ser el título, aunque de nuevo, una gran cantidad de reseñas lo omite. Es importante recalcar que un usuario puede haber realizado múltiples reseñas a un hotel en distintos momentos temporales.

5.2 Preprocesamiento de la base de datos

El preprocesamiento de datos tiene como objetivo transformar los datos brutos originales en un conjunto de datos de mayor calidad y un formato adecuado para su posterior uso. Existe una amplia variedad de técnicas para el preprocesamiento de datos [53], que se suelen agrupar en cuatro categorías.

- Data cleaning o limpieza de datos: orientado a eliminar datos con ruido o incorrectos.
- Data integration: consiste en integrar datos de distintas fuentes de forma homogénea.
- Data transformation: transformación de los datos en formas adecuadas para su preprocesado (técnicas como la normalización).
- Data reduction: trata de reducir el tamaño de los datos mediante agregación o eliminación de características redundantes.

Por último, en este proyecto también es necesario el prepocesamiento del texto que compone cada reseña. En las sección siguiente se explican técnicas de preprocesado generales que son utilizadas para transformar texto en lenguaje natural a un formato más adecuado para su posterior procesamiento informático.

5.2.1 Técnicas de preprocesado de texto

El preprocesado de texto simplemente implica convertir dicho texto a una forma en la que sea predecible y analizable para la tarea a realizar. Una misma técnica de preprocesado puede ser inadecuada para una tarea y adecuada para otra, es tarea del investigador saber qué técnicas utilizar y cuáles no dependiendo del dominio de la tarea. A continuación se relatan algunas de las técnicas más comunes en el preprocesado de textos en lenguaje natural [54], [55].

Capitalización:
Los textos normalmente contienen una variedad de letras mayúsculas para indicar el inicio de una frase o enfatizar los nombres propios. La aproximación más común es reducir todo a minúsculas por simplificación, aunque esto puede ocasionar problemas con las palabras que tienen diferentes significados en mayúsculas y minúsculas. Así por ejemplo el acrónimo TIC, Tecnologías de la Información y de las Comunicaciones, tiene un significado distinto de tic, que es un gesto involuntario.

```
Original: ``Me dio un tic estudiando TIC."

Procesado: ``me dio un tic estudiando tic."
```

Figura 5.2: Ejemplo de transformación a minúsculas.

Eliminación de stop-words o palabras vacías:
La mayoría de las palabras utilizadas en las frases, como los artículos y preprosiciones, no poseen significado por sí mismos sino que conectan sujetos, verbos etc. Estas palabras suelen eliminarse.

```
Original: Le dije que me diera la carta.

Procesado: dije diera carta.
```

Figura 5.3: Ejemplo de eliminación de palabras vacías.

Eliminación de puntuación y caracteres especiales:
Normalmente los signos de puntuación y caracteres especiales no tienen ningún significado intrínseco. Sin embargo esto depende mucho del contexto. Por ejemplo, un signo de exclamación "!" puede implicar descontento o entusiasmo, o en Twitter, una arroba "@" y una almohadilla "#" tienen significados especiales, representando usuarios y hashtags.

```
Original: ``@O'Neil: #AllLivesMatter!"

Procesado: ``ONeil AllLivesMatter"
```

Figura 5.4: Ejemplo de eliminación de puntuación y caracteres especiales.

Tokenización:
Este es el proceso de separar cada párrafo en frases y frases en palabras. Normalmente se usan los signos de puntuación y los espacios para delimitarlos.

```
Original: ``Me apetecen churros, ¿y a ti?"

Procesado: [``Me", ``apetecen", ``churros", ``,", ``¿",
            ``y", ``a", ``ti", ``?"]
```

Figura 5.5: Ejemplo de tokenización.

Truncación y lematización

El objetivo del truncación (stemming en inglés) y la lematización es encontrar la forma base de una palabra, de la que se extrae el significado central de la misma. El uso de estas técnicas está extendido debido a que la mayoría del NLP se centra en el significado de los textos. Difieren tanto en el proceso a seguir como el resultado final. La truncación es un proceso heurístico en el que se eliminan los morfemas de las palabras para encontrar la raíz o lexema, la cual no siempre es una palabra. Por otro lado, la lematización es un proceso más refinado y costoso, con el uso de un vocabulario y análisis morfológico de las palabras, y tiene como resultado final el lema, la forma canónica de la palabra base, que es la que aparece en el diccionario. Por ejemplo, dada la palabra "libros", mediante el uso de truncación se llegaría al lexema "libr", mientras que utilizando lematización se obtendría el lema "libro".

5.3 Representación densa del lenguaje de las reseñas

En esta sección se explica el funcionamiento básico del algoritmo Doc2vec, utilizado en este proyecto para la representación densa del lenguaje de las reseñas, así como su predecesor Word2vec. También se analizan diversos aspectos relacionados con Doc2vec de especial relevancia en este proyecto: el tamaño de la DB necesaria para el algoritmo Doc2vec, el preprocesado de texto óptimo que deben tener las entradas al algoritmo y las métricas de evaluación para medir su rendimiento.

Lenguaje natural y aprendizaje automático

En el aprendizaje automático o machine learning, el primer obstáculo que se encuentra es la representación de los datos que el programa debe aprender. En el caso de variables numéricas, como por ejemplo el precio de un producto, la representación es directa, pero las variables categóricas, como el nombre de dicho producto, presentan un problema, ya que muchos algoritmos no pueden trabajar con estas variables directamente.

Una solución consiste en representar cada categoría como un número entero, método conocido como label encoding. Sin embargo, los números enteros tienen una clara relación de orden, y los algoritmos de aprendizaje automático pueden aprenderla, por lo que, en casos en que las variables no tienen esta relación, una posibilidad es la utilización del llamado one-hot encoding. En esta codificación se asigna un vector binario de longitud igual al número de categorías, con un valor "1" en la posición del

entero asignado en label encoding, y un "0" en el resto (ver figura 5.6).

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Figura 5.6: Integer encoding y one hot encoding de variables categóricas [3].

$$\begin{pmatrix} the \\ cat \\ sat \\ on \\ the \\ mat \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figura 5.7: One hot encoding de una frase en lenguaje natural [4].

Para el análisis de lenguaje natural, esta codificación se vuelve altamente ineficiente, puesto que cada palabra es en sí una categoría. En la figura 5.7 se puede ver la representación de una frase de seis palabras en una matriz de 6x5, siendo el 5 el tamaño del vocabulario (una palabra se repite, "the"). Sin embargo, en aplicaciones reales se utilizan vocabularios mucho mayores, por ejemplo de 10.000 palabras. Esto significa que una red neuronal que quiera aprender dicho vocabulario, necesitará 10.000 nodos de entrada. Además con esta representación desaparece el contexto entre palabras, por ejemplo, las palabras "comida" y "hambre" suelen aparecer juntas en la misma frase. Otro problema del one hot encoding consiste en su compatibilidad con las redes neuronales, ya que la mayoría de estas están pensadas para trabajar con números reales como entradas, y no con valores discretos.

Una solución a todos estos problemas es utilizar la representación densa del lenguaje, en la cual cada palabra es representada como un vector numérico, mediante algoritmos como Word2vec que se explica a continuación.

5.3.1 Word2Vec

Word2vec [27] es un algoritmo de representación densa del lenguaje que proyecta palabras a un espacio vectorial de baja dimensionalidad utilizando una red neuronal de una única capa oculta. El resultado es una serie de vectores llamados Word-vectors para cada palabra, donde las analogías entre palabras tienen su traducción en las posiciones de sus vectores asociados, como se muestra en la figura 5.8.

Existen dos implementaciones de Word2vec llamadas "Skip-gram" y "Continuous bag of words", de las se explicará su funcionamiento básico.

Skip-gram

La red neuronal que entrena este algoritmo tiene como entrada una capa del tamaño del vocabulario, una capa oculta de mucha menor dimensionalidad, llamada

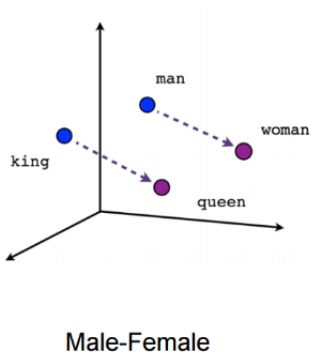

Figura 5.8: Analogías en el espacio vectorial, rey es a reina lo que hombre a mujer [5].

embedding, y una capa de salida del mismo tamaño de la capa de entrada (ver figura 5.10). El objetivo de dicha red es, dada una palabra en formato one-hot a la entrada, predecir probabilidad de que cada una de las otras palabras del vocabulario sea su vecina.

Para representar el contexto de una palabra (las palabras que son cercanas a la misma), este algoritmo se basa en los llamados Grams, que son grupos de n palabras, siendo n el tamaño de ventana alrededor de una palabra central que se incluye en dicho grupo (ver figura 5.9). La parte skip de los skip-grams se refiere al número de palabras de la ventana que finalmente se escoge, llamado skip-window. Así para el 5-gram “the quick brown fox jumps” con skip-window de tamaño 2, la palabra central del gram es “brown”, y de las palabras del contexto en la ventana (“the”, “quick”, “fox”, “jump”), se escogerán dos palabras para hacer el skip-gram final (por ejemplo: “the”, “jump”)

Se entrena la red usando los Skip-grams de las frases del corpus (conjunto de frases) como Mini-batch de entrada, actualizándose los pesos de la red tras cada Mini -batch. Del resultado final se quita la capa de salida, quedando una red que transforma palabras en representación one-hot a un vector de mucha menor dimensionalidad. Así, en el ejemplo de la Figura 5.10, se reduce de un vector de 10.000 dimensiones a uno de 300.

Este algoritmo funciona bien con pocos datos, y representa bien palabras raras (con pocas ocurrencias en el corpus).

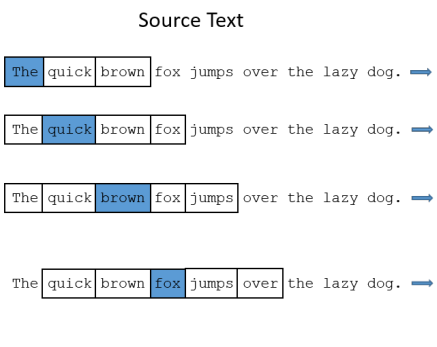

Figura 5.9: Ejemplo de skip-grams [6].

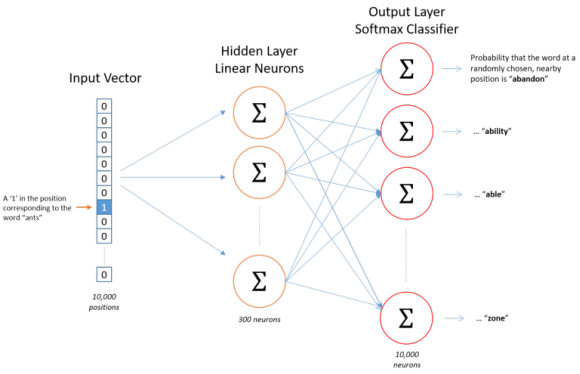

Figura 5.10: Red neuronal de una capa usada en Skip-gram [7].

Continuous bag of words

Este algoritmo es muy similar al anterior, utilizando también una red neuronal de una sola capa oculta, pero con el objetivo contrario: se predice una palabra dado el conjunto de palabras de su contexto (puede verse una comparativa de la estructura de ambos algoritmos en la figura 5.11). En este algoritmo se crea una ventana deslizante alrededor de la palabra actual, y se alimenta la red con la representación one-hot de las palabras de dicha ventana. El algoritmo utiliza la media de los vectores de las palabras para predecir la palabra actual. Después de entrenar, se quita la capa de entrada y se da la vuelta a la red, acabando una red idéntica a la red final de Skip-gram.

Este algoritmo se entrena mucho más rápido que Skip-Gram, y es más adecuado para predecir palabras frecuentes.

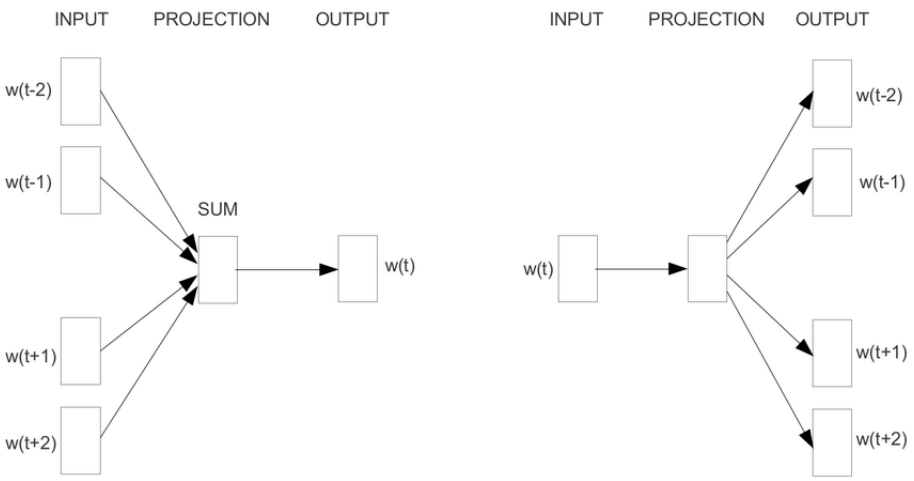

Figura 5.11: Comparativa de los dos algoritmos Word2Vec[8].

Mejoras a word2vec

En las secciones anteriores se ha descrito brevemente el funcionamiento básico de los algoritmos Skip-gram y Continuous bag of words. En ambos casos, los mismos autores de los algoritmos implementaron mejoras de los mismos para conseguir reducir el tiempo computacional: Hierarchical Softmax, Negative Sampling y Subsampling[56], [57].

Hierarchical softmax

El algoritmo Word2Vec Skip-Gram básico tiene una complejidad del tamaño del vocabulario $\mathcal{O}(V)$, debido a la capa softmax de salida que activa cada uno de sus nodos como último paso de la cada computación de la red. El método hierarchical softmax reduce la complejidad al logaritmo en base dos del vocabulario:

$$\mathcal{O}(V) \rightarrow \mathcal{O}(\log_2 V)$$

Esta reducción dramática de la complejidad se debe a que cada vector de salida se determina por una estructura de árbol binario, donde cada hoja representa la probabilidad de una palabra (ver figura 5.12). Se puede llegar a cada palabra desde la raíz por los nodos internos, que representan la función de probabilidad del camino.

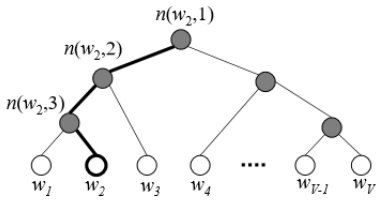

Figura 5.12: Estructura de árbol de hierarchical softmax [9].

Negative sampling

El negative sampling es una manera de entrenar usando ejemplos en lugar de usar todo el corpus, de manera similar al famoso stochastic batch descent. La idea es convertir el problema de clasificación multinomial en una clasificación binaria: para cada muestra de entrenamiento, se alimenta la red con un ejemplo verdadero (una palabra central y palabras de su ventana) y un número de ejemplos falsos (la palabra central y palabras aleatorias del corpus). Aprendiendo a distinguir entre ambos, el algoritmo aprende los Word-vectors en mucho menos tiempo.

Subsampling

El subsampling consiste en eliminar de los documentos de forma aleatoria aquellas palabras que aparecen con mayor frecuencia, con el objetivo de reducir el tiempo de entrenamiento y combatir el desequilibrio entre la frecuencia de estas palabras y las más raras. Este procedimiento es muy similar a la eliminación de palabras vacías y signos de puntuación, ya que estos suelen ser los términos más repetidos, sin embargo mantiene dos diferencias: estas palabras no se eliminan en todos los documentos y no se extraen del análisis lingüistico, si no de la estadística de los textos a analizar. Este último punto marca una diferencia importante, ya que dependiendo del área de trabajo, estos términos pueden cambiar.

5.3.2 Doc2Vec

Word2vec proyecta palabras a un espacio vectorial de baja dimensionalidad, creando Word-vectors. Doc2vec [28] es una extensión simple a este algoritmo para proyectar párrafos o documentos (conjuntos de frases) a otro espacio vectorial de baja dimensionalidad, creando Paragraph-vectors. Existe una extensión para cada uno de los algoritmos de Word2vec: Skip-grams y Continuous bag of words.

PV-DBOW

El algoritmo PV-DBOW (Distributed Bag of Words version of Paragraph-Vector) es una extensión del algoritmo Skip-gram. Mientras que en Skip-gram se predicen las palabras del contexto de cada gram, en PV-DBOW se predicen las palabras del contexto de cada documento. Este algoritmo es más rápido que su predecesor y consume menos memoria, ya que no guarda word-vectors.

PV-DM

El algoritmo PV-DM (Distributed Memory version of Paragraph-Vector) es una extensión del algoritmo Continuous Bag of Words. La idea básica es añadir un vector extra al entrenamiento de todas las palabras de un documento, un identificador del párrafo. De esta manera al entrenar los Word-vectors de un documento, también se entrena un vector representando el propio documento.

Este algoritmo es usualmente superior a PV-DBOW.

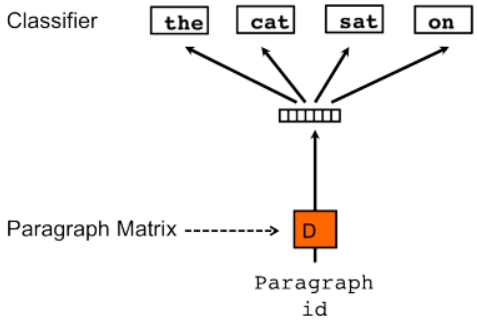

Figura 5.13: Algoritmo PV-DBOW [10].

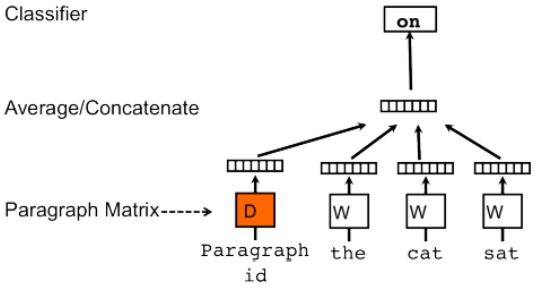

Figura 5.14: Algoritmo PV-DM [10].

5.3.3 Modificación de Doc2Vec en el proyecto

Así como Doc2vec en su versión PV-DM es una extensión sobre Word2vec en la que, además de entrenarse vectores para las distintas frases, se entrena un vector representando cada documento, en este proyecto se realizó una extensión similar sobre Doc2vec. Se añadió al corpus un documento compuesto de todas las reseñas de cada hotel y cada usuario. Así se obtuvo, además de un vector por cada reseña, un vector que represente a cada usuario y cada hotel.
Este método contribuye a que Doc2vec aprenda la similaridad entre los documentos pertenecientes a un mismo hotel/usuario, lo cual es el objetivo final de este algoritmo, demostrando así que se han generalizado las características de las reseñas de cada hotel y usuario.

Puede verse una representación de este algoritmo en la figura 5.15.

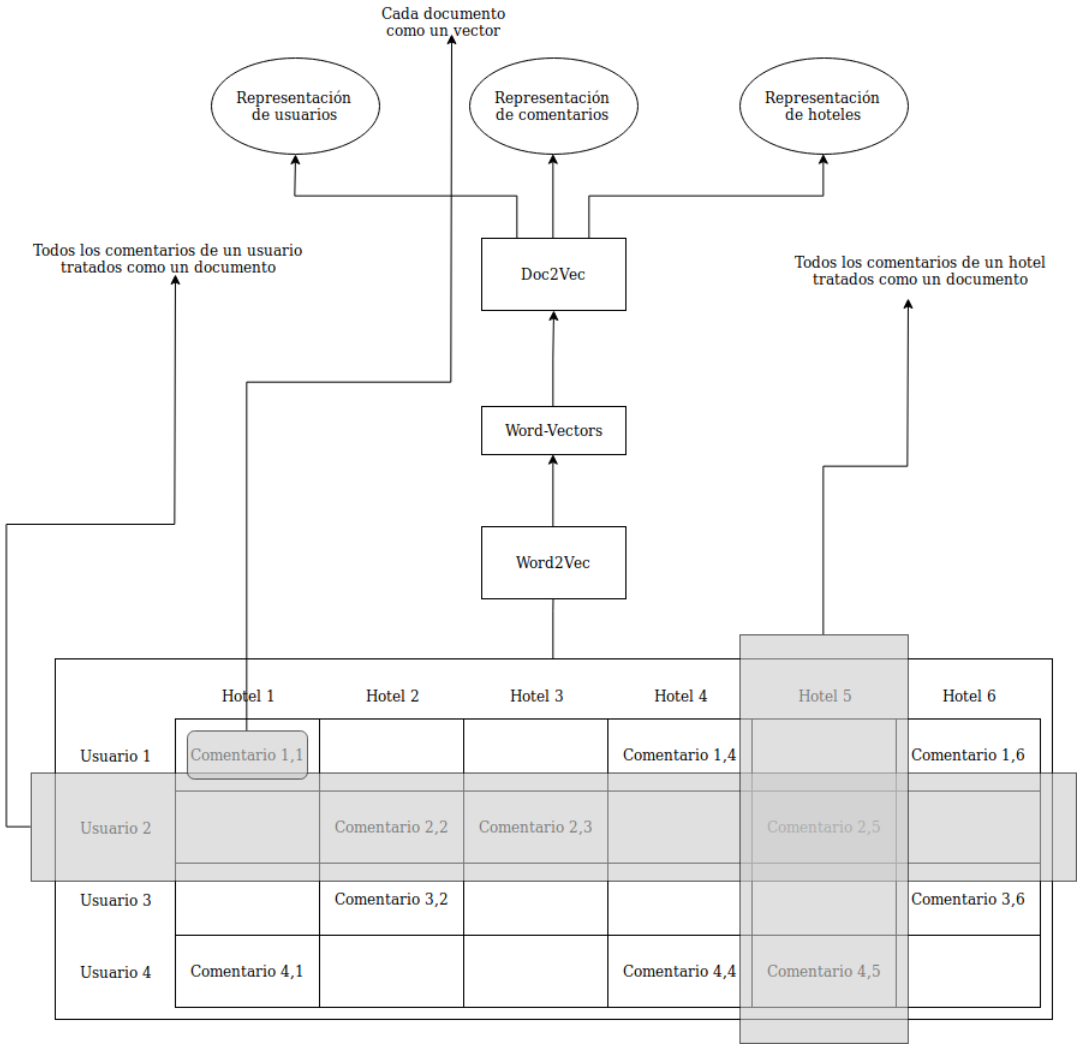

Figura 5.15: Algoritmo de aprendizaje de representación de comentarios, hoteles y usuarios.

5.3.4 Word2vec y Doc2vec, métricas de evaluación

Para medir la similitud entre los vectores de Doc2vec/Word2vec es una técnica estándar el uso de la similitud coseno.

La similitud coseno es una medida de similitud entre dos vectores distintos de cero en un espacio que posee producto interior. Evalúa el coseno del ángulo entre ellos. Es igual a 1 si el ángulo entre ambos vectores es 0, es decir, si apuntan a la misma dirección. En caso de apuntar al sentido contrario, el valor del coseno es -1. Así, el intervalo de resultados de la similitud coseno se encuentra el intervalo cerrado [-1,1]. Esta métrica suele aplicarse en minería de textos para obtener una relación de semejanza entre ellos [58].

Para el uso de la similitud coseno como métrica de evaluación para embeddings existen diferentes métodos. En los papers originales de Word2vec [27], [56] se utilizan analogías entre grupos de palabras con significados semánticos similares. Por ejemplo, dado "Alemania" : "Berlín" :: "Francia" : ?, se resuelve encontrando el vector x tal que vec(x) es el más cercano por distancia coseno a vec("Berlín") - vec("Alemania") + vec("Francia"). Se considerará un acierto si la respuesta x es "París", y un fallo en caso contrario. Con esta tasa de aciertos y fallos, se obtiene una medida de la precisión del algoritmo.

En Doc2vec [28], [59] se suele utilizar una métrica similar, creando analogías en grupos de tres documentos, dos similares y un tercero no relacionado. Cabe destacar que los grupos de tres escogidos manualmente por los autores obtienen una tasa de acierto sensiblemente mayor a los grupos de tres basados en categorías, en los que los dos documentos similares pertenecen a una misma categoría en un dominio determinado. En los experimentos sobre Wikipedia del paper "Document Embedding with Paragraph Vectors" [59], se observa una precisión de 93% al escoger grupos de tres manualmente, frente a un 78,8% al escoger la similitud de los grupos de tres basándose en que los documentos similares pertenecen a la misma categoría. Esta es una diferencia muy significativa, del 14,2%.

La similitud coseno no es adecuada para su uso directo como métrica de evaluación fuera de analogías en grupos de tres. En este proyecto, en varias pruebas de ejemplo con bases de datos de tamaño pequeño (unos pocos miles de comentarios) se utilizó la similitud coseno entre vectores de los mismos usuarios y vectores de los mismos hoteles, acabando con una similitud media de 0.8/0.9 en un rango [-1,1], pero sin

embargo al realizar pruebas de analogía, el % de error era cercano al 40%. Esto se debe a la distribución de los vectores de Doc2vec en el espacio vectorial. No existe ninguna garantía de que los vectores ocupen todo el espacio vectorial y se distribuyan en todas las dimensiones, sino que pueden agruparse. Es decir, pueden ser muy similares entre ellos, siendo aún más similares a aquellos con quienes guardan relación.

5.3.5 Doc2vec y tamaño de bases de datos

El paper original de doc2vec [28] evalúa el algoritmo en tres bases de datos:

- **Stanford Sentiment Treebank**: ∼12.000 frases de reseñas de películas (que fueron seccionadas en ∼240.000 fragmentos de unas cuantas frases cada una.
- **IMDB Dataset**: 100.000 reseñas de películas (usualmente de unos cuantos cientos de palabras por reseña).
- **Search-result snippet paragraphs**: 10.000.000 párrafos, conseguidos del top 10 resultados de búsquedas de Google para cada una de los 1.000.000 consultas más comunes.

Un paper subsiguiente [59] aplica el algoritmo para descubrir relaciones entre categorías en las bases de datos:

- **Wikipedia**: ∼4.500.000 artículos de Wikipedia.
- **Arxiv**: ∼890.000 papers académicos extraídos de PDF.

Es decir, los corpus utilizados en estos primeros papers de doc2vec variaban de cientos de miles a millones de documentos, con tamaños de documento variables entre unas pocas frases a artículos de miles de palabras. Se puede inferir que al menos unos cientos de miles de documentos son necesarios para lograr un buen desempeño con doc2vec, y que en general, los resultados son mejores si los documentos constan de al menos dos o tres frases.

5.3.6 Doc2Vec y preprocesado de texto

Mientras que todas las técnicas explicadas en la sección 5.2.1 son comunes en los esfuerzos de distintos autores para su uso en NLP, las prácticas para Word2Vec y Doc2Vec suelen ser diferentes. Por ejemplo, en el paper original de Word2Vec [27] y sus evaluaciones no se menciona ningún tipo de stemming/lematización o eliminación de palabras vacías, y se retiene la puntuación como tokens representando

palabras en sí mismas. Las palabras incluidas en el set de tres millones de Word-vectors pre-entrenados a partir de palabras de Google News [60] tampoco contienen stemming/lematización e incluyen palabras vacías y mayúsculas. El paper original de Doc2Vec [28] no hace ninguna referencia a un preprocesado extra de sus textos antes de utilizarlos.

Es posible que la influencia de palabras vacías y puntuación sea diferente (y en este caso, positiva) en Word2Vec/Doc2Vec, en constraste con otras formas de NLP. Por ejemplo, con respecto a los Word-vectors, estos tokens pueden ser una señal útil al alinear palabras que se utilizan comúnmente con ciertas palabras vacías, o con algún cambio en las frases, y por tanto indicar algún aspecto de similitud relevante. En algunas revisiones de Word2Vec/Doc2Vec sí se encuentran ciertas formas de preprocesado, por ejemplo en el paper de Lison y Kutuzov [61] se infiere que, como mínimo, la eliminación de palabras vacías sí puede ser beneficiosa para el algoritmo Doc2Vec.

5.4 Optimización de hiperparámetros

Debido a la amplia cantidad de hiperparámetros de los que hace uso el algoritmo Doc2vec, se utilizó un algoritmo de optimización de hiperparámetros para elegir los más adecuados. En esta sección se explica el funcionamiento de algunos tipos de estos algoritmos, así como el Tree-Structured Parzen Stimator, utilizado en el proyecto.

Durante la creación de algoritmos de aprendizaje automático, existen una serie de parámetros que este aprende forma automática mediante el entrenamiento para alcanzar un rendimiento óptimo, como por ejemplo los pesos de las neuronas de una red neuronal. Sin embargo, antes de empezar a entrenar estos algoritmos, se escogen una serie de parámetros llamados hiperparámetros, que son por ejemplo cuántas capas tiene dicha red, cuántas neuronas por capa etc. La elección de hiperparámetros finalmente es la que decide el rendimiento del algoritmo, por lo que es, en sí misma, un problema de optimización.

La implementación más simple de un optimizador de hiperparámetros es la llamada grid search, que es una simple búsqueda por todo el espacio de hiperparámetros. Es decir, se utilizan todas las combinaciones posibles y se evalúa su rendimiento una a una. Esta aproximación es increíblemente ineficiente, en primer lugar porque tiene un coste computacional muy alto, y en segundo lugar porque desaprovecha el hecho

de que ciertos parámetros tienen un impacto muy bajo en el desempeño final de los algoritmos, con lo que debería invertirse más tiempo en optimizar el resto de parámetros.

Una implementación subsiguiente es random search [62], en la que se definen un número de iteraciones para la búsqueda de hiperparámetros y la distribución estadística de los mismos, que se muestreará de forma aleatoria. Con esta aproximación el tiempo de búsqueda se reduce drásticamente, y se aprovecha el hecho de que ciertos parámetros tienen mayor importancia que otros. Sin embargo, este algoritmo presenta fallos evidentes: los valores de mayor importancia deben conocerse de antemano y el muestreo de parámetros es aleatorio, con lo que, si se halla el mejor resultado, será por azar.

Con el paso del tiempo, han surgido diversos algoritmos para la optimización automática de hiperparámetros, ofreciendo alternativas mucho mejores a random search, tanto en tiempo de computación como en efectividad. En este proyecto se utilizó la optimización bayesiana de hiperparámetros y en concreto, el Tree-Structured Parzen Stimator (TPE).

5.4.1 Optimización bayesiana

La optimización bayesiana de hiperparámetros es un término acuñado por Jonas Movckus [63] en relación a su trabajo en métodos de optimización global en los años 70. La estrategia base de este conjunto de algoritmos es tratar la optimización parámetros como una función aleatoria y poner una distribución de probabilidad encima. Esta distribución captura las creencias de los expertos en el desempeño de la función a optimizar. Después de hacer una serie de evaluaciones, que se tratan como datos, estas distribuciones son actualizadas para formar nueva distribución probabilística que determina el siguiente set de parámetros a probar. De esta manera, con el tiempo, el algoritmo aprende a moverse paulatinamente hacia el subespacio de la configuración inicial de parámetros que contiene el mejor rendimiento de la función a evaluar.

5.4.2 Tree-Structured Parzen Stimator

Los TPE [64] son un tipo de algoritmos de optimización bayesiana que tienen una estructura de árbol para la búsqueda en el espacio de configuración de hiperparámetros, lo cual implica que se puede utilizar condicionalidad en la elección de los mismos. Esto quiere decir que las variables de las hojas del árbol (por ejemplo, el número de

neuronas de la segunda capa de una red neuronal) sólo se definen cuando las variables de los nodos (en este caso, el número de capas de la red) toman valores concretos. Así es posible definir espacios de búsqueda anidados de manera simple e incluso múltiples algoritmos con una misma ejecución del programa.

5.5 Predicción de vectores hotel/usuario

Una vez transformadas las reseñas de texto de la base de datos a vectores mediante representación densa del lenguaje, se obtiene una base de datos muy similar al esquema inicial de un RS (ver figura 1.3). El siguiente paso en el proyecto es la predicción de los vectores de hotel y usuario faltantes en la base de datos. En esta sección se explican diversos algoritmos utilizados en RS para la recomendación, así como el algoritmo que se adaptó para su uso en este proyecto, el Neural Collaborative Filtering.

Como se adelantaba en el capítulo de introducción (ver capítulo 1) en este proyecto se utiliza un RS de filtrado colaborativo. Este método de filtrado se basa en encontrar usuarios con las mismas preferencias utilizando las opiniones de los mismos sobre una serie de productos (normalmente, en forma de puntuación numérica). Su finalidad es, como la de todos los RS, ofrecer recomendaciones personalizadas a los usuarios. Existen dos categorías principales de filtrado colaborativo:

- **Basados en vecindad:** se centran en encontrar la relación entre productos o entre usuarios. Esta aproximación evalúa las preferencias de un usuario por un producto basándose en las puntuaciones de los productos vecinos por el mismo usuario. Los vecinos de un producto son otros productos que tienen a tener una puntuación similar si los evalúa el mismo usuario.

- **Basados en factores latentes:** explica las puntuaciones al caracterizar ambos usuarios y productos en múltiples factores inferidos por el patrón de las puntuaciones. Por ejemplo, en la recomendación de música, los factores latentes podrían medir dimensiones precisas como la longitud de las canciones, el género musical... O dimensiones que se desconocen. Para los usuarios, cada factor mide cuánto le gusta a un usuario los productos que tienen un alto valor en el factor latente correspondiente.

En este proyecto existe un incentivo para el uso de un algoritmo de RS basado en factores latentes, ya que como se explica en la sección 5.3.3, se utilizará el algoritmo Doc2vec para aprender un vector que represente a cada usuario y hotel, siendo este

vector la representación de dichos factores latentes.

Uno de los algoritmos basados en factores latentes más extendidos es la Factorización de Matrices [65]. Esta caracteriza a productos y usuarios utilizando vectores de factores inferidos por el patrón de las puntuaciones, aprendiendo así una representación de baja dimensionalidad para cada usuario y producto (ver figura 5.16). Para ello se usa una técnica similar a la que utiliza Word2Vec para mapear los vectores one-hot correspondientes a las palabras del corpus a evaluar con su representación vectorial: el uso de embeddings. En el caso de predecir un único valor, se puede utiliza una red neuronal con dos entradas, una para usuarios y otra para productos. Cada una de estas entradas pasa por una capa oculta de embedding y se utiliza el producto escalar de la salida como método para reducir los dos vectores de los embeddings a un único valor, que será la valoración predicha de un usuario a un producto. Usualmente se normalizan los valores de entrada para usar la llamada Factorización No Negativa de Matrices, que es idéntica a la Factorización de Matrices pero añadiendo restricción de no negatividad a los embeddings aprendidos, lo cual facilita el aprendizaje.

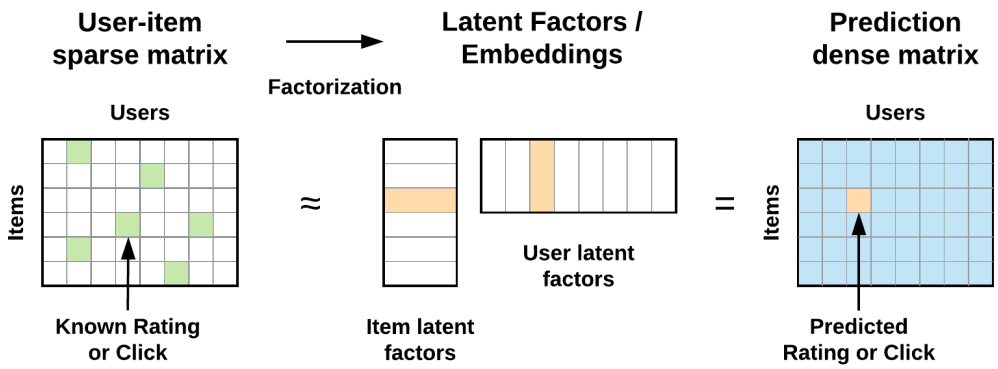

Figura 5.16: Factorización de Matrices [11].

Existe gran variedad de algoritmos de aprendizaje profundo (deep learning en inglés) aplicadas a RS [66]. En este proyecto se estudió la aplicación de diversos algoritmos dependiendo de los resultados obtenidos. Estos algoritmos se explican a continuación.

5.5.1 Recomendación basada en Perceptrón Multicapa

Un Perceptrón Multicapa (MLP por sus siglas en inglés) es una red neuronal concisa pero efectiva que ha demostrado ser capaz de aproximar cualquier función medible a un grado deseado de precisión [67]. Los MLP se pueden utilizar para añadir transformaciones no lineales a las aproximaciones de RS existentes. Este punto es importante en el proyecto, ya que las distintas dimensiones de los vectores que se intentan predecir (los Paragraph-vectors dados por Doc2vec) tienen relaciones no lineales entre sí, no es posible predecir cada dimensión por separado. Dos ejemplos ilustrativos de MLP son el Neural Collaborative Filtering (NCF) y la Neural Network Matrix Factorization (NNMF)[66].

El esquema de un NCF [12] se muestra en la figura 5.17. En la capa de entrada, los usuarios y productos están codificados con one-hot encoding para ser posteriormente mapeados a una capa de embeddings (factores latentes de cada usuario y producto). Las capas subsiguientes pueden ser cualquier tipo de conexiones neurales, como un MLP. Gracias a la conexión y no linealidad de estas capas, el modelo es capaz de estimar las complejas interacciones entre el espacio de factores latentes de usuarios y productos.

Una aproximación subsiguiente, que añade aún más no linealidad a este modelo, es la NNMF [12] que se muestra en la figura 5.18. Esta incluye un módulo MLP a la aproximación anterior, concatenando la salida de ambos módulos.

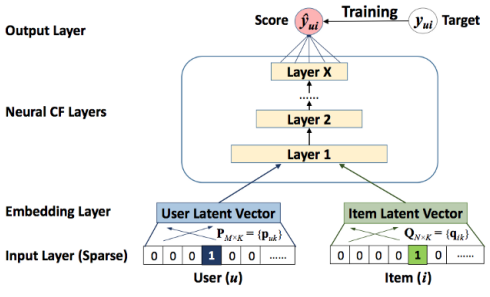

Figura 5.17: Ilustración de NCF, un MLP de Filtrado Colaborativo [12].

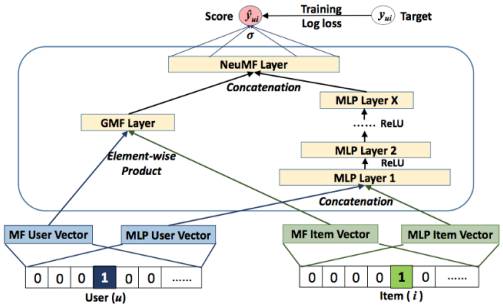

Figura 5.18: Ilustración de NNMF, un MLP de Filtrado Colaborativo [12].

5.5.2 Algoritmo de predicción usado en el proyecto

Como se explicó en la sección 5.5, existe una gran diversidad de aproximaciones para la predicción en RS. En este proyecto, existe un incentivo para el uso de un algoritmo de RS de filtrado colaborativo basado en factores latentes. Estos factores latentes (también llamados embeddings) son una representación vectorial de baja dimensionalidad que representa características de cada producto y usuario de un RS.

Debido a la utilización del algoritmo Doc2Vec para aprender la representación vectorial de cada reseña de texto al mismo tiempo que se aprende la representación de cada usuario y de cada hotel (explicado en la sección 5.3.3), ya se han obtenido los vectores de factores latentes de cada hotel y usuario, lo cual es el punto clave de cualquier algoritmo de RS basado en factores latentes.

Se hizo una adaptación de uno de estos algoritmos, llamado NCF (se puede ver en la figura 5.17). Al tener de antemano los embeddings de usuarios y hoteles, el algoritmo se simplifica a una regresión múltiple y multivariante implementada por una red neuronal. Se modifica también la salida del NCF en esta regresión, para que en lugar de condensar la salida en un único valor numérico usando el producto escalar, la salida sea un vector de con las mismas N dimensiones que los vectores de reseñas que se intentan predecir.

Para medir la pérdida de la red, es decir, la diferencia entre el vector a predecir y el predicho, se utilizó la distancia coseno, valor complementario a la similitud coseno ya utilizada en el proyecto (ver sección 5.3.4). La distancia coseno da resultados en el rango [0,2], siendo 0 la menor distancia posible entre vectores (vectores iguales).

Puede verse la estructura de la red en la figura 5.19.

Se planteó el uso de otros algoritmos, pero tras los resultados arrojados por este, se consideró que el objetivo del proyecto ya se había cumplido.

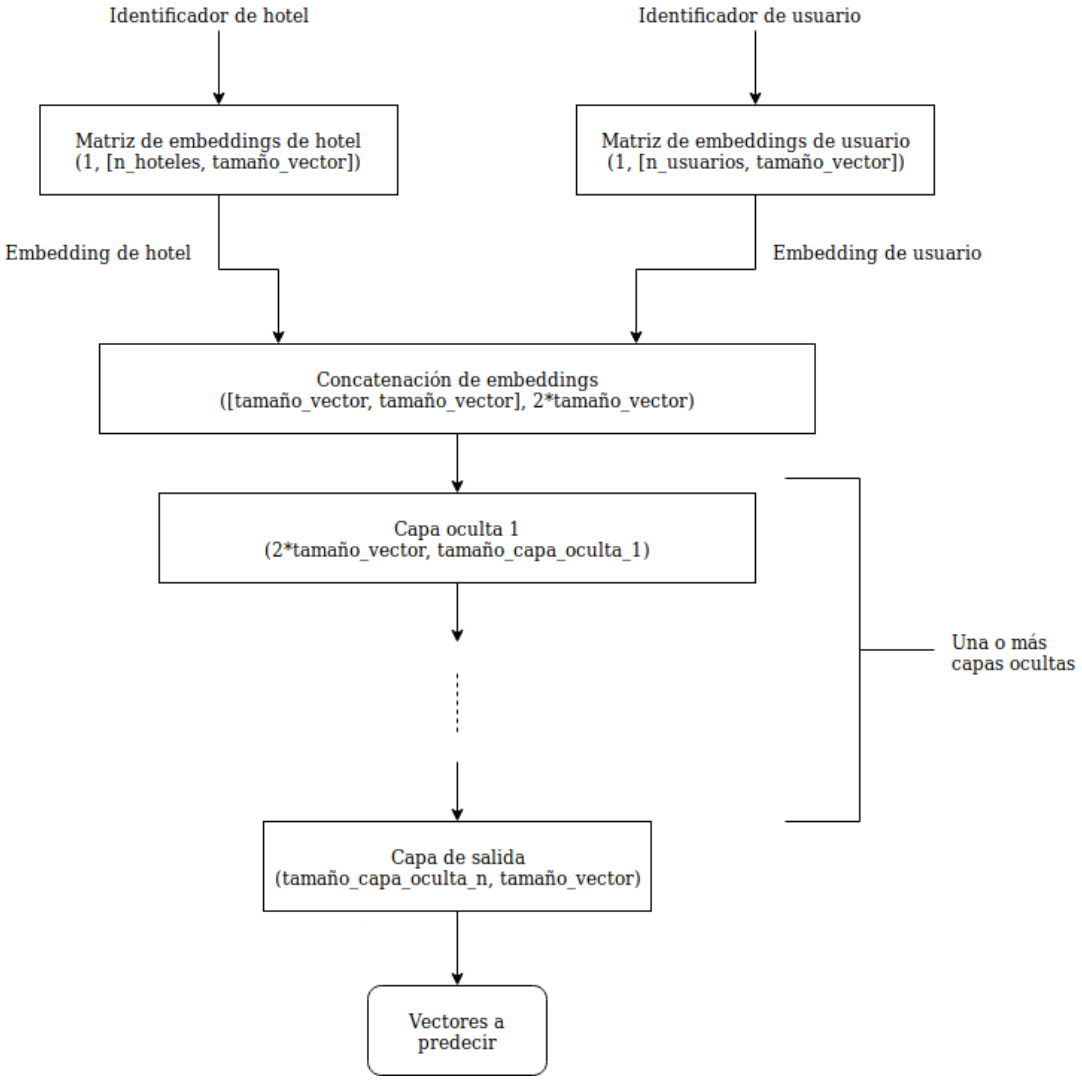

Figura 5.19: Adaptación de NCF usada en el proyecto.

Capítulo 6

Desarrollo

En este apartado se describe cada uno de los pasos llevados a cabo para conseguir los resultados finales del proyecto a partir de la base de datos inicial, aplicando los algoritmos descritos en el capítulo "Materiales y métodos" (capítulo 5) y midiendo su rendimiento. Se comentan los resultados intermedios de cada paso.

Conviene recordar el esquema del proyecto (ver figura 1.4): en primer lugar se obtiene la base de datos y se preprocesa, en segundo lugar se obtiene una representación densa del lenguaje de las reseñas de la base de datos mediante el uso del algoritmo Doc2Vec, y por último, se aplica un algoritmo de aprendizaje profundo para predecir las reseñas de usuarios a hoteles que no han visitado, y se miden los resultados.

El lenguaje de programación utilizado para el desarrollo del proyecto fue Python (ver sección 4.1). Es un lenguaje ideal para una metodología de desarrollo ágil (como es Scrum, la metodología usada en este proyecto), debido a su sencillez, facilidad de lectura y su extensa colección de bibliotecas.

6.1 Preprocesamiento de la base de datos

Como se detallaba en la sección 5.1, la base de datos usada en el proyecto tiene numerosos problemas, como campos faltantes e inconsistencias en el formato. Para el desarrollo de este proyecto se necesitan fundamentalmente tres elementos: identificadores de hotel, identificadores de usuario y el texto de reseña que deja cada usuario a un hotel. Se eliminó el resto de datos no necesarios (Data Reduction) para acelerar el posterior uso de la base de datos.

Fue necesario el uso de técnicas de Data Cleaning para eliminar inconsistencias.

Debido a los problemas con los identificadores numéricos existentes de hoteles, se utilizaron los nombres unívocos de hotel como base para establecer nuevos identificadores numéricos (necesarios para el posterior uso de estos datos en con algoritmos de aprendizaje automático). Se obviaron aquellos hoteles sin nombre, al no poder distinguirlos individualmente. De igual manera, para identificar a cada usuario se utilizó su nombre de usuario unívoco y se estableció un identificador numérico único para cada uno. Aquellos usuarios con reseñas anónimas, cuyo nombre quedaba representado por "A TripAdvisor Member" o "Posted by an anonymous TripAdvisor user", fueron ignorados al no poder identificarlos. Existen usuarios que realizaron más de una reseña en un mismo hotel, en diferentes periodos de tiempo. En esos casos se obvió la reseña más antigua, al entenderse que la más reciente reflejaría mejor información sobre el estado actual del hotel.

Para aumentar la sencillez del código y la capacidad de paralelización al recorrer la base de datos, se reunieron las reseñas de cada hotel en un único fichero Json. Además se eliminaron los hoteles y usuarios con pocas reseñas, ya que es necesario un mínimo de reseñas para poder generalizar las características de usuarios y hoteles con Doc2Vec (explicado en sección 5.3.5). Este número se determinó analizando la base de datos tras el preprocesado, y se explica en la sección subsiguiente.

En cuanto al texto de las reseñas, no existe un consenso en qué técnicas de preprocesamiento son las correctas para el algoritmo Doc2Vec, o si es siquiera necesario el uso de cualquier técnica (como se adelantaba en en la sección 5.3.6). Debido a esto, en el proyecto se han aplicado dos conjuntos de técnicas de preprocesamiento a la base de datos, creando dos bases de datos diferentes con las que se evaluará el rendimiento final del proyecto. En la primera base de datos se aplicó tokenización y se eliminaron los comentarios muy pequeños (>50 palabras), al considerarse que no tienen suficiente información. La segunda base de datos añadió a estas técnicas: eliminación de palabras vacías, puntuación y signos especiales. Se planteó la creación de una tercera base de datos, añadiendo técnicas más avanzadas de preprocesamiento como stemming si los resultados finalmente indicaban que el algoritmo se beneficiaba del uso de preprocesamiento.

6.1.1 Resultados del preprocesamiento

Después del preprocesamiento básico, se obtuvo un conjunto de datos con aproximadamente $\sim$8000 hoteles, con una media de 114 reseñas, $\sim$650.000 usuarios, con una

media de 1 reseña cada uno. Cada comentario tiene una longitud media de 220 palabras (contando palabras vacías, signos de puntuación y caracteres especiales). De las ∼5.200.000.000 posibles combinaciones de hotel y usuario hay un total de ∼900.000 comentarios y ∼200.000.000 de palabras, con un vocabulario de ∼750.000 palabras.

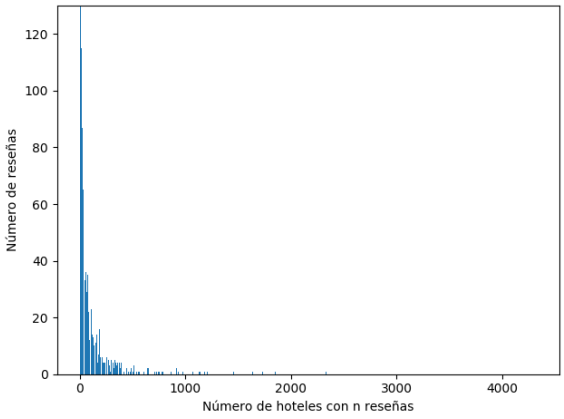

Figura 6.1: Relación de número de hoteles y su número de reseñas.

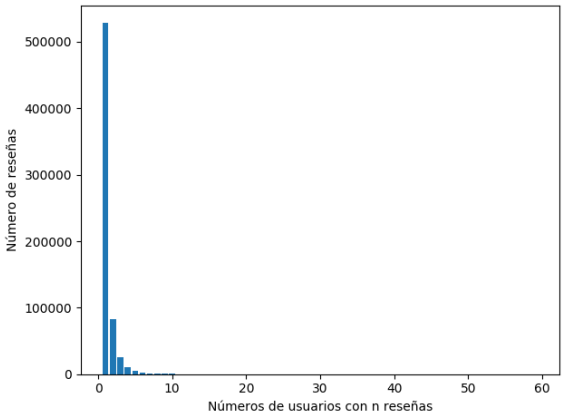

Figura 6.2: Relación de número de usuarios y su número de reseñas.

Se puede observar por las estadísticas anteriores y las figuras 6.1 y 6.2 que casi la totalidad de los usuarios han dejado su opinión una única vez en TripAdvisor, y de igual manera, la gran mayoría de hoteles sólo han recibido una reseña, aunando un pequeño conjunto de hoteles la mayoría de las mismas. Esto por desgracia es un problema del dominio, y debido a que se necesita un mínimo de reseñas para generalizar las características de cada hotel y usuario, se tuvo que prescindir de gran parte de estas.

Con el segundo método de preprocesado, la distribución de reseñas se mantuvo prácticamente idéntica, pero debido a las mayores restricciones impuestas (eliminación de palabras vacías y puntuación, manteniendo una longitud mayor a 50 palabras), la cantidad se reduce. Esta base de datos contiene aproximadamente ∼8000 hoteles, con una media de 83 reseñas, ∼490.000 usuarios, con una media de 1 reseña cada uno. Cada reseña tiene una longitud media de 120 palabras. Al compararlo con la media de palabras en la otra base de datos, se infiere que aproximadamente 100 de cada 220 palabras de un comentario son palabras vacías y signos de puntuación. De las ∼3.800.000.000 posibles combinaciones de hotel y usuario hay un total de ∼650.000 reseñas y ∼78.000.000 de palabras, con un vocabulario de ∼290.000 palabras.

En la tabla 6.1 se pueden observar las reseñas con ambos bases de datos en fun-

Reseñas por número mínimo de comentarios de hotel y usuario		
Mínimo de reseñas de hotel y usuario	Total de reseñas de la primera DB	Total de reseñas de la segunda DB
1	900.000	650.000
2	370.000	250.000
3	200.000	130.000
4	115.000	70.000
5	70.000	35.000

Tabla 6.1: Reseñas por número mínimo de comentarios de hotel y usuario.

ción del número mínimo de comentarios de cada hotel y usuario. Para balancear la necesidad de grandes cantidades de reseñas con un número mínimo de comentarios por hotel y usuario suficiente para extraer las características comunes de sus reseñas, finalmente se escogieron aquellos con al menos 3 reseñas, ya que aumentar este número resultaría en el uso bases de datos excesivamente alejadas del mínimo en literatura (ver sección 5.3.5), bajando incluso de cien mil reseñas. Cabe destacar, que de aquellos usuarios con un mínimo de 3 reseñas, más de la mitad tienen al menos 4 reseñas, y de estos, más de la mitad tienen al menos 5 o más; es decir, no todos los hoteles y usuarios tienen exactamente 3 reseñas.

6.2 Representación densa del lenguaje

Una vez se han preparado las bases de datos, el siguiente paso es conseguir una representación densa de los textos de las reseñas para posteriormente poder predecirlas.

El algoritmo de representación densa utilizado en el proyecto fue una modificación del algoritmo Doc2Vec que además de aprender la representación vectorial del texto de cada reseña, aprende la representación de cada hotel y usuario, explicada en la sección 5.3.3.

Implementación:
Para la implementación del algoritmo se utilizó la biblioteca Gensim [43], que implementa dicho algoritmo como una clase, la cual se encapsuló en un wrapper propio para extender su funcionalidad. Dicha clase entrena una instancia interna de Doc2vec

al pasarle un iterador con los documentos del corpus así como una serie de parámetros, de los que se destacan los más útiles en la tabla 6.2.

Parámetro	Explicación	Tipo de dato y rango útil
dm	Versión del algoritmo doc2vec a utilizar	1 para PV-DM, PV-DBOW en caso contrario
vector_size	Dimensionalidad de los Paragraph-Vectors	Número entero en rango $[1, \infty)$
window	Tamaño de ventana	Número entero en rango $[1, \infty)$
learning_rate	Ratio de aprendizaje	Float > 0
min_count	Apariciones mínimas en el corpus para considerarse palabra	Número entero en rango $[0, \infty)$
epochs	Número de iteraciones del algoritmo	Entero en rango $[1, \infty)$
hs	Uso de modo Hierarchical Softmax o Negative Exponent	1 para Hierarchical Softmax, Negative exponent en caso contrario si negative no es 0
negative	Número de palabras utilizadas en Negative Sampling	Número entero en rango $[0, \infty)$
negative_exponent	Exponente usado para la distribución negativa	Float en rango [0, 1)
dm_mean	Utilizar suma o media de los word-vectors del contexto	0 para la suma, 1 para la media
sample	Rango para la frecuencia con la que las palabras más frecuentes se reducen de forma aleatoria	Rango útil = (0,1exp -5)

Tabla 6.2: Variables a utilizar en el algoritmo doc2vec.

Gensim requiere que cada texto que le pasa el iterador vaya acompañado de un identificador, llamado tag, permitiendo el uso de un mismo tag varias veces (se entrenan los dos documentos concatenados) e incluso de varios tags (el documento se entrena dos veces, una por tag). De esta manera, se suplió por cada reseña un tag único así como un tag que identifica al hotel y otra al usuario que corresponde, logrando que se entrene un texto por cada hotel y usuario compuesto por la concatenación de todas las reseñas. La clase Doc2vec de la biblioteca Gensim también contiene una gran cantidad de funciones para interacturar con la instancia de Doc2vec que han sido útiles en el proyecto:

- **infer_vector**: usado para inferir el Paragraph-vector de un documento que no se encontraba en el entrenamiento. Para ello, entrena la red Doc2vec de forma temporal como si este documento estuviese en el corpus.
- **most_similar**: devuelve una lista de tags y vectores correspondientes a los n Paragraph-vectors más similares al dado, computado por similaridad coseno.
- **load_model** y **save_model**: utilizados para cargar y guardar el objeto Doc2vec a varios ficheros, respectivamente.

Una vez entrenada la red del algoritmo Doc2vec, se obtiene una representación vectorial del texto de cada reseña (así como una de cada hotel/usuario). Se trata de vectores con N dimensiones (parámetro escogido en entrenamiento mediante la variable vector_size), con valores en rango [-1,1] y de tipo float32. Debido a que todos los valores de los vectores están en el mismo rango, no es necesario normalizarlos.

Métrica de evaluación:
Como métrica de evaluación para analizar la calidad de la representación densa se utilizaron las analogías en grupos de tres explicadas en la sección 5.3.4. Se hicieron dos grupos de tres por cada vector dado por Doc2vec. Un primer grupo contiene el vector original, un vector similar que proviene de una reseña del mismo hotel, y uno no relacionado. El segundo grupo contiene el vector original, otro del mismo usuario, y un tercero no relacionado. Se midió la similitud coseno entre el vector original y el similar, y entre el original y el no relacionado. En caso de que la primera sea mayor, se consideró como positivo, en caso contrario, como negativo, y se obtuvo finalmente un porcentaje de acierto y error basado en el número de positivos y negativos. Es decir, se midió la cantidad de veces que el algoritmo es capaz de predecir adecuadamente la similitud entre dos textos pertenecientes al mismo hotel o usuario después de haber

sido entrenado con estos textos, de manera similar al paper "Document Embedding with Paragraph Vectors" [59].

Optimización de hiperparámetros:
Debido a la gran variedad de hiperparámetros entre los que escoger para utilizar Doc2vec (visibles en la tabla 6.2), la gran diversidad de valores para los mismos utilizados en los papers de los autores de Doc2vec [28], [59] y recomendaciones de autores sucesivos [20] en combinación con el alto coste computacional y temporal de utilizar este algoritmo, lo hacen un candidato ideal para utilizar un algoritmo de **optimización de hiperparámetros**. Cabe destacar que otros autores anteriormente [68] también consideraron necesario el uso de un algoritmo de optimización de hiperparámetros, optando por Grid Search (explicado en la sección 5.4), una opción completa pero con un coste computacional muy alto. En este proyecto se utilizó la biblioteca Hyperopt [46], basada en un TPE (explicado en la sección 5.4.2) para la elección automática de hiperparámetros. Esta biblioteca implementa una función de optimización que recibe un diccionario de hiperparámetros con sus distribuciones de probabilidad y una función que minimizar. Esta función de optimización realiza una serie de iteraciones aplicando los hiperparámetros sobre la función a minimizar, siendo las 20 primeras iteraciones aleatorias, para después utilizar los datos de estas iteraciones para la búsqueda de los mejores hiperparámetros. Cabe destacar que todos los datos relevantes de la búsqueda están implementados en un objeto serializable, con lo que se pueden guardar utilizando la biblioteca dill [41] o pickle para retomarlos más adelante. Esto último es especialmente útil, debido al largo tiempo de ejecución del algoritmo. El proyecto se ejecutó en los servidores del CESGA, que funcionan por un sistema de colas de prioridad con tiempo límite de ejecución por cada tarea enviada (explicado en detalle en la sección 4.4), por lo que fue necesario particionar la ejecución de proyecto en diferentes segmentos.

6.2.1 Experimento de representación densa sobre las dos bases de datos

Como se explica en la sección 6.1, se utilizaron dos bases de datos, una con un preprocesamiento ligero y otra con un preprocesamiento más agresivo, para comprobar si Doc2vec se beneficia de dicho preprocesamiento. En resumen: sobre cada base de datos, se aplicó una adaptación de Doc2vec, pasando cada reseña de texto además de como un único documento de texto las diferentes reseñas de cada hotel y las de cada usuario. Se utilizó un algoritmo de optimización bayesiana (biblioteca Hyperopt) para la optimización de hiperparámetros de Doc2vec, probando los diferentes hiperpará-

metros de la figura 6.3 durante 100 iteraciones en ambas bases de datos. Se evaluó el rendimiento de cada iteración del algoritmo con las analogías en grupos de tres sobre todo el conjunto de entrenamiento, siendo el porcentaje de error de esta medida el valor que intenta minimizar el algoritmo de optimización.

```
import math
import hyperopt.hp as hp

d2v_hyperparameter_dictionary={
    # Se añade +1 al extremo del rango porque si no la función range lo obviaría
    "vector_size": hp.choice("vector_size", range(50, 300+1, 50)),
    "dm": hp.choice("dm", (0,1)),
    "window": hp.choice("window", (3, 5, 7, 12, 15)),
    "learning_rate": hp.loguniform("learning_rate", math.log(0.0025), math.log(0.25)),
    "min_count": hp.choice("min_count", (2, 5, 10, 20)),
    "epochs": hp.choice("epochs", range(10, 20+1,5)),
    "hs": hp.choice("hs", [

        {
            "type": 0,
            "negative": hp.choice("negative", range(5, 20 + 1, 5)),
            "ns_exponent": hp.uniform("ns_exponent", -1.4, 1.4)
        },
        {
            "type": 1
        }
    ]),
    "dm_mean": hp.choice("dm_mean", (0,1)),
    "sample": hp.choice("sample", [
        {
            "sample_value": 0,
        },
        {
            "sample_value": hp.uniform("sample_value", 0, 1e-5),
        }
    ]),
}
```

Figura 6.3: Hiperparámetros en forma de árbol utilizados para Doc2vec.

La tabla 6.3 contiene los resultados de la analogía en grupos de tres sobre la mejor iteración de Doc2vec en ambas dos bases de datos. Se puede observar que la base de datos con un mayor preprocesado obtuvo un 7.1% más de error que la base de datos con un preprocesado mínimo, en la que se incluían palabras vacías y signos de puntuación, confirmando la sospecha inicial de la sección 5.3.6 de que en Doc2vec, el preprocesamiento tradicional de texto no es efectivo. Conviene destacar la utilidad del optimizador de hiperparámetros usado, ya que en los primeros experimentos realizados con parámetros seleccionados manualmente, el error era cercano al 30%.

Base de datos	Porcentaje de error
DB con preprocesado medio	24,2%
DB con preprocesado mínimo	17,3%

Tabla 6.3: Resultados del experimento en las dos bases de datos.

6.2.2 Experimento de representación densa con ampliación del espacio de hiperparámetros

Debido a que la instancia que alcanzó el error mínimo en el experimento utilizó el valor máximo en dos hiperparámetros, el tamaño de vector (vector_size) y la cantidad de iteraciones (epochs), con un valor de 300 y 20, respectivamente, se realizó otro experimento aumentando ambos valores y utilizando únicamente la base de datos con mínimo preprocesado, al haberse demostrado que esta es la mejor para su uso con Doc2vec. Ambos parámetros son de los más influyentes en relación al tiempo de ejecución.

En el caso del tamaño de vector, la medida más grande en literatura [59] es de 10.000, pero se utiliza para aprender documentos muy grandes, como son los artículos de Wikipedia. Estudiando el resto de la literatura [28], [20], aunque el rango normal es de 100 a 300, existen varios casos con un valor de 300 a 400. Se utilizará un valor de 500 para el nuevo tamaño máximo de vector, para aumentar este rango y asegurar que la convergencia del algoritmo no se ve limitada por este factor.

Respecto a las iteraciones; habitualmente se utiliza un rango de 10 a 20, sin embargo, debido a que la cantidad de datos que se suple en este proyecto a Doc2vec está cercana al mínimo de la literatura, es posible que un número mayor de iteraciones ayude a generalizar mejor. Debido a que este parámetro es el que más influye en el tiempo de ejecución (ya que indica el número de veces que el algoritmo itera sobre el corpus), subirlo de forma indiscriminada puede hacer que cada iteración dure días, un coste inasumible en el proyecto. Se tomó un valor relativamente alto, de 50 iteraciones máximas, con el cual una iteración del optimizador de hipérparámetros puede tardar dos horas como máximo en ejecutarse. Debido a que la ejecución del experimento se lleva a cabo en los servidores del CESGA mediante un sistema de colas de prioridad, el tiempo final de ejecución fue mayor de dos semanas, con lo que, para futuras revisiones de este proyecto se recomienda encarecidamente el uso de un servidor dedicado.

Resultados:
Como se puede observar en la tabla 6.4, con la ampliación de los límites en hiperparámetros, el error se reduce en un 2.8%. Los parámetros finales utilizados se pueden ver en la tabla 6.5. Cabe destacar que el tamaño de vector (vector_size) no llegó al valor máximo establecido de 500, si no que se quedó en 450, con lo que la decisión de aumentar el rango a 500 en lugar de los 400 de la literatura fue acertada. Sin embargo, respecto al número de iteraciones, inesperadamente sí se alcanzó el máximo de 50, debido probablemente al tamaño relativamente pequeño de la base de datos. Se planteó el uso de un mayor valor para las iteraciones para mejorar el resultado, pero debido al muy elevado tiempo de ejecución se descartó. En futuras revisiones del proyecto se recomienda aumentar el tamaño de las bases de datos y no restringir las iteraciones máximas del algoritmo.

Base de datos	Porcentaje de error
Hiperparámetros iniciales	17,3%
Hiperparámetros ampliados	14,5%

Tabla 6.4: Resultados del experimento según hiperparámetros.

En la figura 6.4 se puede observar la pérdida de la función de optimización de hiperparámetros (es decir, el error de la instancia de Doc2vec entrenada) a través de las 100 iteraciones. En las primeras 20 iteraciones, donde los hiperparámetros escogidos por el algoritmo de optimización son aleatorios, se encuentran grandes fluctuaciones y los valores de pérdida más altos. El algoritmo alcanza su mínimo alrededor de la iteración 63, y puede observarse que a continuación intenta explorar el espacio de hiperparámetros cercano, sin éxito, al conseguir valores de pérdida ligeramente superiores, para finalmente seguir probando otros parámetros. A continuación en la figura 6.5 se ilustra un gráfico de dispersión que compara el uso de los diferentes hiperparámetros en el experimento con los hiperparámetros ampliados, y toma un color de amarillo a violeta en función del error (siendo el violeta el menor error). Puede observarse que el uso de la implementación de Doc2vec PV-DBOW es siempre mejor a PV-DM en este proyecto, cuando normalmente en literatura se da el caso contrario.

Se hará un análisis de estos resultados en el capítulo 7.

Parámetro	Explicación
dm	0, uso de PV-DBOW
vector_size	450
window	5
learning_rate	0,0087 aprox.
min_count	5
epochs	50
hs	1, uso de Negative Exponent
negative	10
negative_exponent	0,2367 aprox.
dm_mean	0, suma de los word vectors
sample	0, sin subsampling

Tabla 6.5: Hiperparámetros finales de Doc2vec.

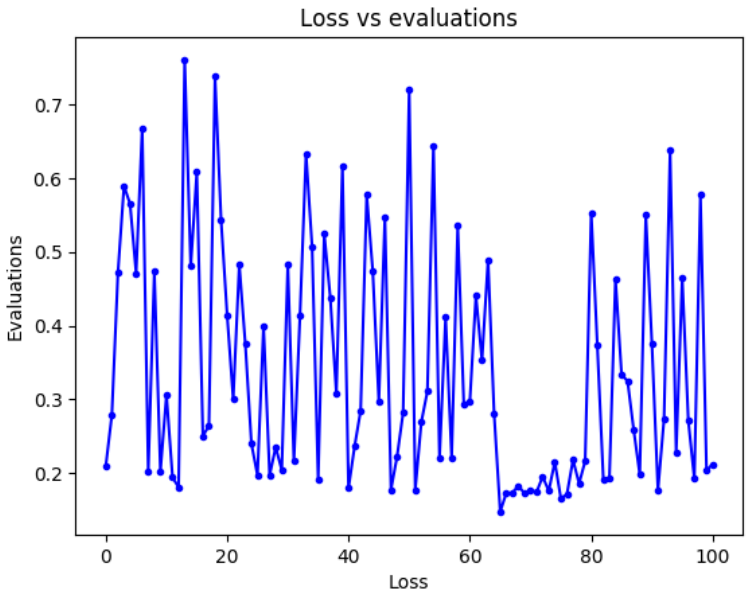

Figura 6.4: Pérdida de la función de optimización de hiperparámetros a través de 100 iteraciones.

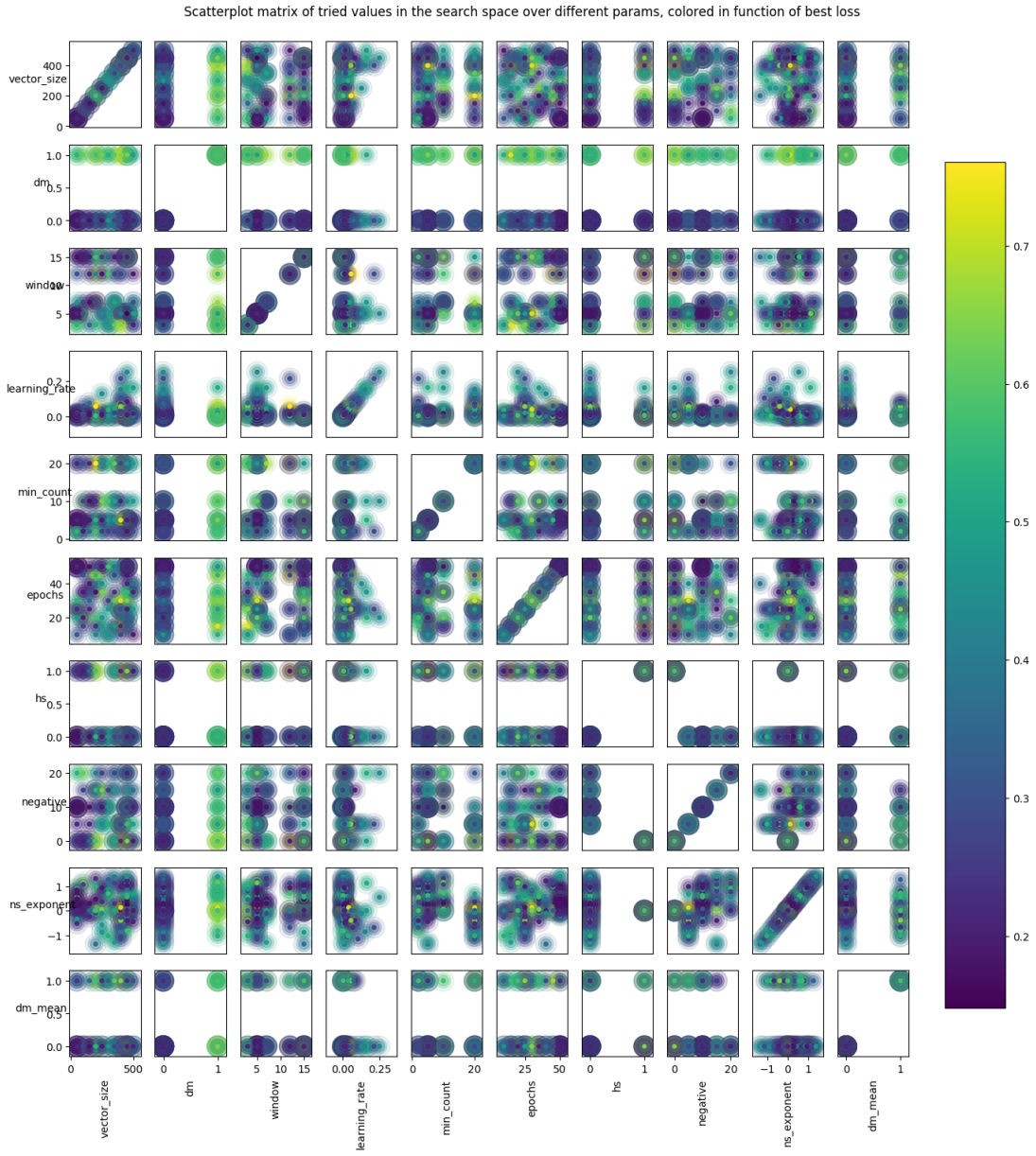

Figura 6.5: Gráfico de dispersión de los valores de hiperparámetros de Doc2vec en relación a la pérdida.

6.3 Predicción de vectores

Una vez transformados los comentarios de la base de datos en vectores, se obtiene una base de datos muy similar al esquema inicial de un RS 1.3. El siguiente paso en el proyecto es la predicción de los vectores de hotel y usuario faltantes en la base de datos.

Una primera aproximación podría haber sido utilizar un algoritmo de Factorización de Matrices para predecir cada una de las dimensiones de los vectores. Sin embargo, además de ser una aproximación inadmisiblemente lenta (ya que requeriría entrenar tantas redes como dimensiones tenga un vector), esta no conservaría la relación que existe entre las dimensiones de los vectores. Es necesaria una aproximación que sea capaz de generalizar todas las dimensiones de los vectores a la vez.

Finalmente en el proyecto se utilizó una modificación de la red neuronal NCF, explicada en la sección 5.5.2. Esta aproximación toma ventaja del hecho de que ya se tienen representaciones vectoriales para cada hotel y usuario. Puede verse la estructura de la red utilizada en la figura 5.19.

Hiperparámetros de la red:
Las dimensiones de los embeddings de usuario y hotel, la entrada de la red y salida de la red ya están definidas. Tanto los embeddings como la salida tienen el mismo tamaño que los vectores entrenados por el algoritmo Doc2vec. La red tiene dos entradas simultáneas, que son el identificador de hotel y usuario de cada vector. Dichas entradas se utilizan para seleccionar el embedding correspondiente a dicho hotel y usuario dentro de dos matrices, una que contiene los embeddings de hoteles y otra de usuarios. Se utilizó una primera capa para concatenar los embeddings y una o más capas ocultas consecutivas antes de la capa de salida.

Como tamaño de Mini-batch para cada iteración se utilizó 32, siguiendo las recomendaciones de Dominic y Carlo en un reciente paper de investigación [69] donde aseguran que los tamaños entre 2 y 32 dan de forma constante mejores resultados.

Buscando que la regresión no sobreentrene (que se adapte demasiado bien a los datos de entrenamiento en lugar de generalizar sus características), se dividieron los datos suministrados a la red en dos subconjuntos, uno de entrenamiento y otro de validación, con el 90% y el 10% de los datos totales respectivamente, y se aplicó la técnica de early stopping monitorizando el valor de pérdida en el conjunto de valida-

ción. Es decir, se parará el entrenamiento una vez el error de validación comience a aumentar de forma sistemática, punto en el que la red estará perdiendo capacidad de abstracción y sobreentrenando.

Esta red neuronal se implementó utilizando la bibioteca Keras [44], que cuenta con una API de alto nivel para la creación de redes neuronales, con el backend de Tensorflow y utilizando múltiples GPU para su entrenamiento.

Evaluación de rendimiento:
Para la evaluación del rendimiento de la red se utilizó la técnica de validación cruzada k-folds. Esta consiste en dividir en k partes o "folds" la base de datos. Se escoge 1 fold para test y el resto (k-1) se utiliza para entrenar un algoritmo. El desempeño del algoritmo luego es probado utilizando el fold de test. Después se repite este proceso k veces, escogiendo cada vez un fold distinto para test, y se hace una media con el error en test. En el caso de este proyecto, se utilizó la ténica k-folds con k=5, es decir, 5 particiones iguales. En cada iteración se escogió 1 fold para test, y con las 4 folds restantes, se entrenó una instancia de Doc2vec y una regresión. Cabe destacar que para la regresión, finalmente se subdividirán los datos una vez más, en conjunto de entrenamiento y validación. Se muestra la división de los datos en una iteración de k-folds en la figura 6.6.

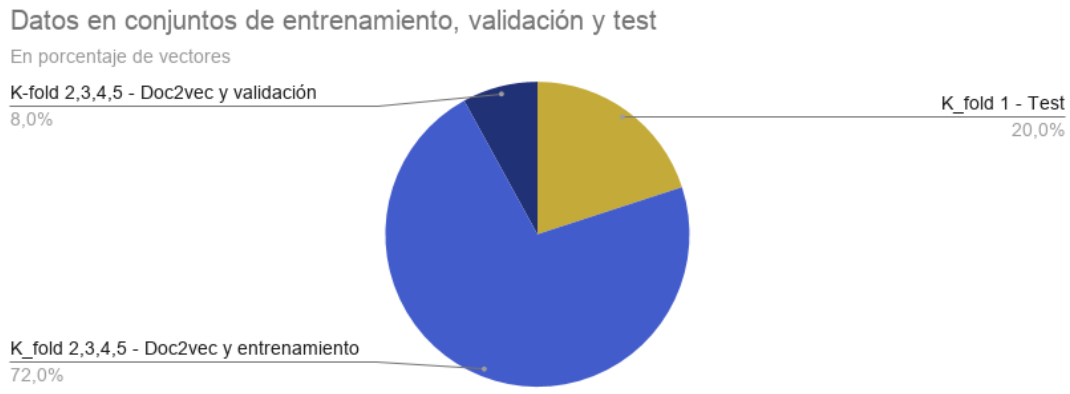

Figura 6.6: Distribución de los datos en conjuntos de entrenamiento, validación y test.

Para medir el rendimiento en el k-fold de test, se retomaron las analogías en grupos de tres usadas para evaluar Doc2vec. Para cada reseña de un usuario y hotel en test, se extrajeron dos vectores diferentes. Para el primer vector, se infirió su representación vectorial al alimentar la regresión con el identificador de ese usuario y hotel

y se escogió como valor final el vector visto en entrenamiento más cercano al inferido (medido por similaridad coseno). El segundo vector se extrajo pasando el texto de la reseña por la función infer_vector de la librería Gensim, que permite obtener la auténtica representación vectorial que la reseña de texto hubiese tenido de haber estado en el conjunto de entrenamiento. Dados estos dos vectores se evaluó como positivo si la similitud coseno entre los dos vectores es mayor a la similitud de un vector de otro usuario y hotel de la instancia Doc2vec entrenada, negativo en caso contrario, y se utilizó el número de negativos y positivos para obtener un porcentaje de acierto y error. Es decir, se está comprobando si la regresión predice de forma correcta un vector similar el que habría dado Doc2vec.

Antes de realizar las pruebas de rendimiento de este algoritmo de predicción, se llevó a cabo un pequeño experimento para comprobar la fiabilidad de la inferencia de reseñas con la función infer_vector. Se tomaron pruebas similares a las aplicadas en la evaluación de Doc2vec, con analogías en grupos de tres. Para cada reseña, se coge su vector en Doc2vec, el vector inferido por la función infer_vector del texto de esa reseña, y un tercer vector no relacionado, y se mide la similitud coseno entre ellas, contando como ejemplo positivo si el vector original y el inferido son los más similares, y negativo en caso contrario. El porcentaje de error fue cercano al 0%, de lo que se puede interpretar que la función infer_vector es capaz de reproducir fielmente los vectores dados por Doc2vec.

6.3.1 Predicción de vectores con primera instancia de Doc2vec

Debido al largo tiempo de ejecución del experimento Doc2vec con hiperparámetros ampliados (de varias semanas en total), se hizo en primer lugar la predicción de vectores sobre la instancia anterior.

Dado el diseño de red para regresión (ver figura 5.19), los parámetros a optimizar son: el número de capas ocultas, la cantidad de neuronas de las capas, la inicialización de los pesos de las neuronas, la función de activación de las neuronas de las capas ocultas, la función de optimización de la red, los parámetros de early stopping y la cantidad de iteraciones del entrenamiento.

Estos últimos dos parámetros son simples, se escogió una cantidad muy elevada de iteraciones, que nunca se alcanzaron debido al early stopping. En este caso, si error en validación de la red deja de mejorar durante 10 iteraciones (valor escogido tras experimentación), se parará el entrenamiento y se cogerá el modelo en la iteración con

mejor error de validación.

Se comenzó utilizando una función de optimización sencilla, Descenso de gradientes estocástico, SGD por sus siglas en inglés. Para las capas ocultas se utilizó la popular función de activación ReLU [70], y una simple activación lineal a la salida. El tamaño de los vectores a predecir era de 300 dimensiones, por lo que la entrada a la red consistía de dos vectores de 300 dimensiones, concatenados a una capa de 600 en total. Para empezar se utilizaron dos capas ocultas de 500 y 400 neuronas respectivamente, con los pesos inicializados utilizando una distribución normal glorot [71].

Una vez establecidos todos los parámetros iniciales, se realizó el experimento. Sin embargo, debido a la extrema lentitud del SGD, se cambió la función de optimización a Adam [72] con los mismos parámetros vistos en su paper de presentación. Se trata una extensión al clásico SGD que en los últimos tiempos ha tenido una amplia adopción en deep learning. Este algoritmo adapta automáticamente su ratio de aprendizaje durante el entrenamiento de una forma que mejora los resultados de anteriores extensiones del Descenso de gradiente de características similares: RMSProp y AdaGrad. De esta manera se logra la convergencia de la red en un número mucho menor de iteraciones, como puede verse en las figuras 6.7 y 6.8, donde se comparara la pérdida de entrenamiento con ambas funciones. Así pues, para las pruebas subsiguientes y la elección de hiperparámetros, se prosiguió con Adam.

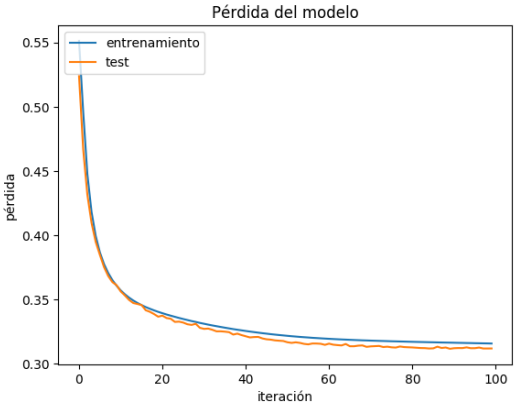

Figura 6.7: Pérdida por iteración de la red usando SGD.

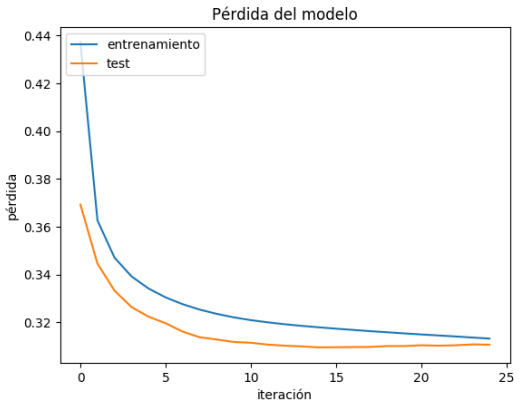

Figura 6.8: Pérdida por iteración de la red usando Adam.

Durante sucesivas ejecuciones del experimento se observó que dos capas de neuronas eran óptimas para esta red, siendo una capa incapaz de converger a la misma

pérdida que dos capas, y tres o más capas covergiendo al mismo error que dos pero en más tiempo. La cantidad de neuronas inicial fue óptima. La función de activación de las capas ocultas mejoró usando PReLU [73], una extensión a ReLU que solventa algunos de sus problemas, como es el acabar con "neuronas muertas", en el que una neurona siempre tiene como salida 0 independientemente de la entrada. Por último, la inicialización de pesos de las neuronas de las capas ocultas final fue una distribución normal He, propuesta en el mismo paper que PReLU [73].

En la tabla 6.6 se presentan los resultados del experimento, con el valor de pérdida en entrenamiento y validación, así como el valor de error de test.

Regresión	Pérdida en entrena-miento rango [0,2]	Pérdida en validación rango [0,2]	Error en test
Primera regresión SGD	0,35	0,34	23,3%
Primera regresión Adam	0,34	0,34	23,5%
Regresión optimizada Adam	0,32	0,31	20,1%

Tabla 6.6: Resultados de diversos experimentos representativos de regresión con validación cruzada.

6.3.2 Predicción de vectores con segunda instancia de Doc2vec

La red Doc2vec con hiperparámetros ampliados da vectores de 450 dimensiones, con lo que como mínimo, era necesario cambiar las dimensiones de los vectores de entrada y salida de la red de regresión a 450. Después de diversas pruebas, este modelo de regresión alcanzó el óptimo adaptando solamente el tamaño de las dos capas ocultas, siendo de 900 y 675 neuronas respectivamente.

En la tabla 6.7 se muestra una comparativa de resultados del experimento anterior utilizando la primera configuración de Doc2vec y del experimento actual utilizando Doc2vec con hiperparámetros ampliados.

Se hará un análisis de estos resultados en el capítulo 7.

Regresión	Pérdida en entrenamiento rango [0,2]	Pérdida en validación rango [0,2]	Error en test
Regresión con primera configuración en Doc2Vec	0,32	0,31	20,1%
Regresión con hiperparámetros ampliados en Doc2Vec	0,28	0,27	18,2%

Tabla 6.7: Mejores resultados de los dos experimentos de predicción de vectores.

Capítulo 7

Resultados finales y análisis

En este capítulo se exponen y analizan los resultados finales obtenidos tras el desarrollo del proyecto.

Conviene de nuevo recordar el esquema del proyecto (ver figura 1.4): en primer lugar se obtiene la base de datos y se preprocesa, en segundo lugar se obtiene una representación densa del lenguaje de las reseñas de la base de datos mediante el uso del algoritmo Doc2vec, y por último, se aplica un algoritmo de aprendizaje profundo para predecir las reseñas de usuarios a hoteles que no han visitado, y se miden los resultados. De todo este proceso, se consideran dos resultados diferentes como resultados finales del proyecto a analizar: la calidad de la representación densa del lenguaje con Doc2vec y la calidad de los vectores y los textos que son finalmente predichos por el sistema.

7.1 Representación densa con Doc2vec

En primer lugar, se analizan los experimentos de representación densa con Doc2vec, en los que se entrenó una instancia de Doc2vec con reseñas de cada usuario y hotel, además de un documento por hotel con todas sus reseñas y otro por usuario. Los resultados se muestran en la tabla 7.1.

Debido a la ausencia de proyectos similares en literatura, resulta difícil evaluar el rendimiento de este trabajo. Los proyectos que utilizan Doc2vec se centran en la categorización de documentos en un conjunto de categorías pequeño y conocido de antemano, como podrían ser los distintos subforos de un foro de internet [20], evaluando el rendimiento mediante el error de clasificación o métricas dadas por el problema, de forma cuantitativa con ejemplos escogidos a mano de documentos similares, y de for-

Base de datos	Porcentaje de error
Hiperparámetros iniciales	17,3%
Hiperparámetros ampliados	14,5%

Tabla 7.1: Resultados del experimento según hiperparámetros

ma cualitativa con analogías semánticas basadas en conocimiento de los documentos del dominio (explicado en la sección 5.3.4). La base de datos utilizada en este proyecto sin embargo contiene una cantidad ingente de categorías (8.000 hoteles y 650.000 usuarios) y un tamaño pequeño (200.000 reseñas) en comparación con la literatura.

En resumen, no parece procedente la comparación directa de los resultados de este experimento con lo anteriormente publicado en literatura.

Un resultado de error en el experimento del 14,5%, y por tanto un acierto de 85,5% puede considerarse cualitativamente aceptable, como muestra de que el sistema está aprendiendo del contenido de los comentarios dado que la mayoría de las veces es capaz de detectar que los comentarios de un usuario/hotel son más similares entre sí que al resto. Si bien la tasa de error es apreciable, esto es comprensible en un problema tan complejo, y teniendo en cuenta el bajo tamaño muestral para algunos hoteles y usuarios y en general el bajo tamaño de la base de datos.

7.2 Predicción de vectores y transformación en texto

En cuanto al experimento de predicción de vectores, en el que se hace validación cruzada para evaluar si una regresión es capaz de predecir los vectores de Doc2vec, los resultados se muestran en la tabla 7.2.

En un principio, un 18,2% de error puede parecer elevado, pero es necesario tener en cuenta que en las pruebas de Doc2vec, para predecir vectores de entrenamiento, se observa un error del 14,5%. Estos dos resultados no son directamente comparables, ya que los experimentos en la predicción de vectores se evalúan sobre una partición de test en lugar de la de entrenamiento, y al usar validación cruzada, se entrenan instancias de Doc2vec con menos reseñas. Sin embargo, esto pone aún más a favor del resultado de 18,2%, pues teniendo más trabas es capaz de obtener un error simi-

Regresión	Pérdida en entrenamiento rango [0,2]	Pérdida en validación rango [0,2]	Error en test
Regresión con primera configuración en Doc2Vec	0,32	0,31	20,1%
Regresión con hiperparámetros ampliados en Doc2Vec	0,28	0,27	18,2%

Tabla 7.2: Mejores resultados de los dos experimentos de predicción de vectores.

lar al que se obtiene en las primeras pruebas con Doc2vec. Se demuestra por tanto que las representaciones densas obtenidas para cada hotel y usuario pueden utilizarse como vectores de factores latentes para un algoritmo de RS basado en factores latentes, como es el caso de NCF, simplificándose a una regresión, al no necesitar obtener de nuevo dichos vectores. Considerando la complejidad del problema junto con las limitaciones del proyecto (base de datos relativamente pequeña, falta de literatura específica que aborde este problema con un método similar), se ha considerado que los resultados son más que aceptables.

Para una evaluación cualitativa del resultado del proyecto, se muestran a continuación ejemplos de reseñas cuyos vectores se demostraron similares durante la validación cruzada. Cabe destacar, como se explicaba en la sección 6.3, que después de predecir un vector mediante regresión, se busca el vector visto en entrenamiento más cercano al predicho, y se escoge el texto del que procede este vector como texto que representa al vector predicho, con lo que este texto siempre es coherente. En futuras iteraciones del proyecto, podría buscarse un método de transformar directamente vectores en comentarios, teniendo en cuenta que entonces sería necesario garantizar la cohesión y coherencia del texto.

```
Hotel: "BEST WESTERN Hotel Artdeco"
Usuario: "Edward L"
Reseña: "It's within walking distance from Termini the main train
    station, but when you have large luggages to carry, you may want
    to find somewhere else. The included breakfast was good. We
    stayed there for only one night before coming home and had to
    catch an early train in the morning , so did not spend a lot of
    time in the breakfast room. However , it was good. Lots of
    choices of tasty baked goods , and some hot stuff e.g . scambled
    egg. The receptionist was friendly but did not offer any extra
    help. Although we did not need any recommendation, but he didn't
    even bother asking. The free Wi-Fi requires a code, which you
    have to ask thereception for. Each equipment - computer, iphone,
    etc . - needs a separate code."
```

Figura 7.1: Reseña original

```
Hotel: "BEST WESTERN Hotel Artdeco"
Usuario: "disgrun"
Reseña: "We stayed at the Art Deco for four nights. It was a good
    choice, as it is located close to Termini station, but not close
    enough to have that 'near the train station' feel to it. The
    reception staff was generally pleasant, and the room we had was
    of a good size and was pretty modern , with satellite TV , an
    electronic safe, and a very good central air conditioning
    system. The bathroom was clean and new, and we were pleasantly
    surprised to find that our bath/shower unit was in fact a shower
    and a jacuzzi bath – but then we found out that the jacuzzi was
    inoperable, which was a pity. The lack of free wireless internet
    was also a pity, as many hotels are now offering this in Europe.
    However, the hotel did offer an internet station in the
    reception area which was free (and fast), and which also had
    free printing facilities (although the printer was pretty slow).
    Breakfast at the hotel was continental, but with some 'hot'
    items such as bacon and eggs as well. The hotel staff was happy
    to call airport taxis and to arrange for taxi pick-ups for us."
```

Figura 7.2: Reseña predicha

En este primer ejemplo de dos reseñas similares, el sistema ha escogido dos reseñas del mismo hotel, por lo que contaría como un resultado positivo en la evaluación del sistema. Puede observarse una serie de características comunes entre ellos: son comentarios principalmente positivos, hablan de la localización (cerca de la estación

de tren), del desayuno, del buen servicio de la recepción...

A continuación se muestra otro ejemplo, de dos reseñas que no pertenecen al mismo hotel o usuario, y que el sistema considera similares, pero que dan un resultado negativo en la evaluación.

```
Hotel: "The Arctic Club Seattle - a DoubleTree by Hilton Hotel"
Usuario: "Amber S"
Reseña: "From the second I walked into the hotel I was greeted with
    warm smiles and a friendly `` hello. '' My fiance and I were
    staying in Seattle for just one night for the annual Seattle
    Firefighter Stair Climb and we were trying to find a nice hotel
    that was somewhat affordable for downtown. This hotel was not
    cheap by any means but it was reasonable compared to the other
    hotels in the area. The room was small but very clean and
    quaint. We were given warm chocolate chip cookies upon arrival
    which I absolutely loved! I really liked the look of the hotel
    and the old fashioned style. The bed was comfortable but the
    pillows were a little too soft. Also, the traffic from outside
    was loud and kept me up all nigh. Suggestion... bring ear plugs.
    Our mini fridge was also making a buzzing noise all night long,
    which didn't help with our sleep either. We didn't spend a lot
    of time in the room or at the hotel but the time we spent was
    pleasant. Always someone opening the door for you and asking if
    we needed any help or had any questions. We will for sure be
    staying at this same hotel next year."
```

Figura 7.3: Reseña original

```
Hotel: "Grand Peninsula Hotel"
Usuario: "Kerry L"
Reseña: "I stayed here for 3 nights in November and had a great
    time. The service was excellent and the staff were great -
    helpful and very friendly. The location was great and the rooms
    were very comfortable. Breakfasf is great too. I would
    definitely return to this hotel."
```

Figura 7.4: Reseña predicha

Este ejemplo es quizá un caso extremo, en el que se observan dos comentarios de dos hoteles y usuarios distintos, de longitudes muy diferentes y que a priori no tienen muchas coincidencias en las palabras que utilizan. Un ser humano puede ver a simple

vista que estos dos comentarios son tremendamente positivos en relación al hotel, e identificar una serie de palabras clave que puede haberlos llevado a ser similares, como sería la palabra “friendly”, sin embargo este proceso resulta mucho más complejo para un algoritmo.

Se consideró aplicar otras técnicas para medir la similitud entre reseñas utilizando los textos de las reseñas en lugar de su representación, pero como se puede observar, es necesario un análisis profundo de todo el texto de las reseñas como el que hace Doc2Vec para poder extraer significado de las mismas. Técnicas que sencillamente miden la similitud entre dos textos sin tener en cuenta el resto de la base de datos no pueden obtener la misma información. En el caso del popular algoritmo BLEU [74] por ejemplo, que mide coincidencias de palabras utilizando ventanas móviles sobre el texto, estos dos comentarios tendrían una similitud muy baja, al no tener apenas palabras en común.

Capítulo 8

Conclusiones y líneas futuras

En este capítulo se exponen las conclusiones obtenidas después de la realización del proyecto junto con posibles mejoras que se podrían aplicar al mismo.

El objetivo de este proyecto era abrir una vía para el mejor aprovechamiento de la información contenida en los comentarios en lenguaje natural que realizan los usuarios de RS, así como el uso de comentarios coherentes en lenguaje natural como resultado último de la recomendación.

Se ha comprobado que la representación densa de reseñas mediante el algoritmo Doc2vec es una manera efectiva de aprender relaciones entre productos y usuarios de un RS, aún con la cantidad mínima de datos mostrada en la literatura. Con este punto, se completa parcialmente el objetivo del proyecto; abrir una vía para el mejor aprovechamiento de la información contenida en los comentarios en NL que realizan los usuarios de RS. También se infiere, a través de la comparativa entre el uso de bases de datos con distintos niveles de preprocesado, que en el uso del algoritmo Doc2vec existe cierta ventaja en mantener palabras vacías, signos de puntuación y caracteres especiales en el corpus del texto a analizar.

Una novedad que surgió debido a los requerimientos del proyecto fue la utilización de un algoritmo de optimización de hiperparámetros, cuyo uso no estaba en la hoja de ruta del proyecto, y demostró sobradamente su utilidad.

Por último, se ha comprobado que, además de ayudar a la robustez de la clasificación de usuarios y hoteles del modelo Doc2Vec, aprender la representación densa para cada usuario y hotel simplifica mucho la tarea de predicción necesaria para estimar

los vectores de comentarios no vistos en entrenamiento, con una implementación clásica de red neuronal profunda que consigue resultados comparables a los del propio experimento con Doc2Vec (aunque cabe destacar que se hicieron 5 modelos diferentes de tamaño 20% más pequeños al original utilizado, con lo que es posible que la comparación directa de resultados no sea del todo precisa).

En resumen, se han satisfecho los objetivos que se perseguían con este proyecto, lo cual no quiere decir que no haya margen de mejora.

Para futuras iteraciones, existen una serie de cambios que podrían utilizarse para mejorar el rendimiento y ampliar la funcionalidad de este proyecto:

- Adquisición de bases de datos mayores para estar a la par con los papers originales de Doc2Vec.
- Uso de una implementación en GPU para el algoritmo Doc2Vec, para mejorar velocidad.
- Combinación de esfuerzos de todos los distintos campos posibles de las reseñas de usuarios: puntuaciones numéricas, comentarios y fotos, siendo este último campo un territorio con menor uso que los comentarios.
- Combinación de la información de las reseñas con otra información típica de RS (por ejemplo ventas de productos, información de los clientes como sexo y edad etc.).
- Traducción inversa, de vectores a frases, en lugar de la utilización del vecino más cercano.
- Uso de técnicas más avanzadas para predicción de vectores.
- Descarte de reseñas poco relevantes para el perfilado usuarios y hoteles (como NARRE [30], explicado en la sección 3.1).
- Empleo de la temporalidad para perfilar los gustos dinámicos de los usuarios (como DER [32], explicado en la sección 3.1).
- Empleo de la temporalidad para dar más peso a las reseñas recientes (como hacen Amazon y Youtube, explicado en la sección 3.2).

Apéndices

Apéndice A

Material adicional

A continuación se muestran los diagramas de planificación (figura A.1 y seguimiento del proyecto (figura A.2) en formato horizontal, para una mejor lectura.

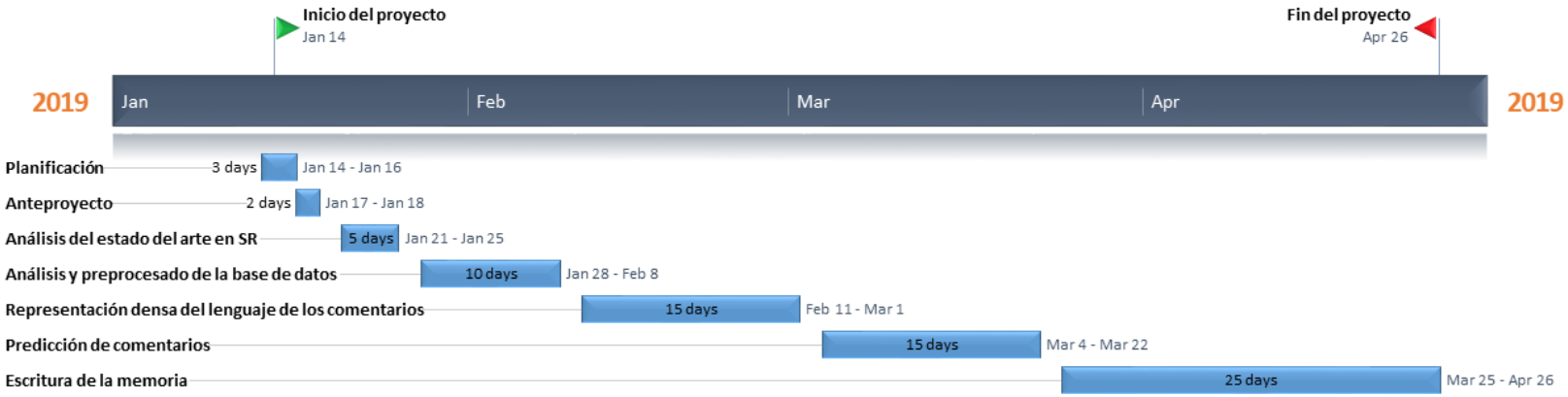

Figura A.1: Planificación inicial del proyecto

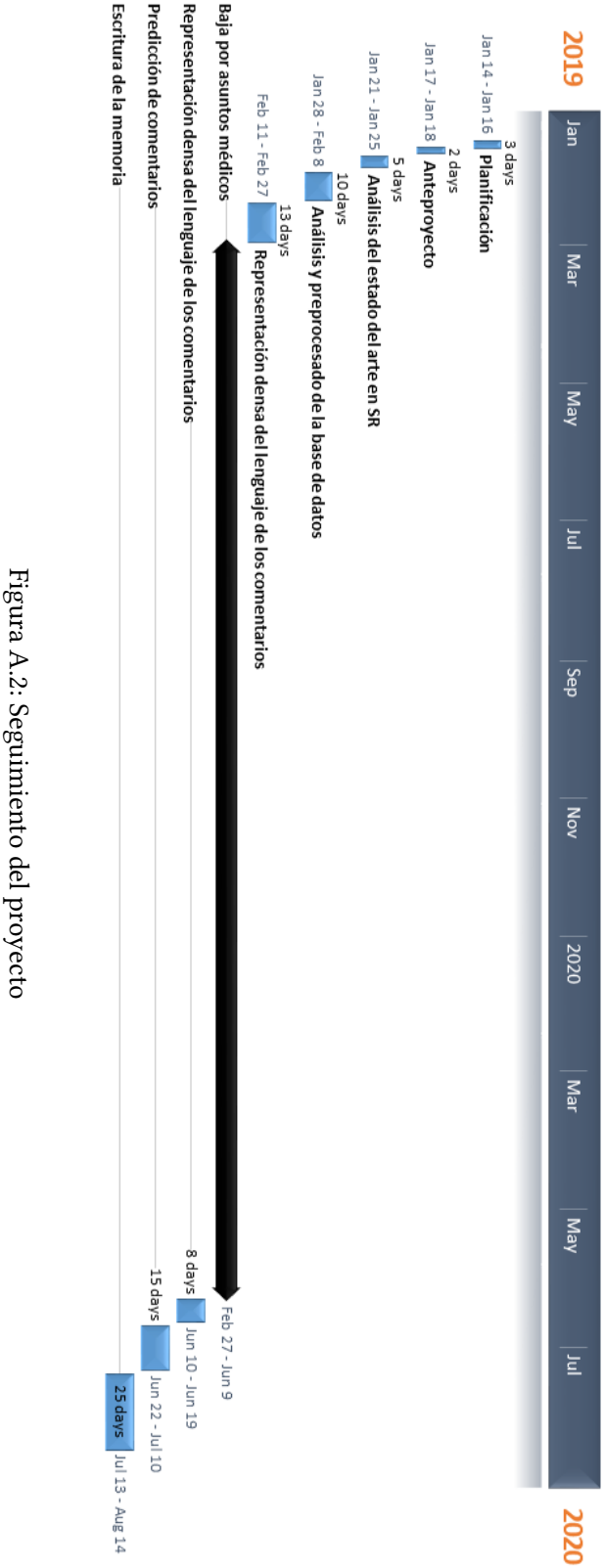

Figura A.2: Seguimiento del proyecto

Lista de acrónimos

CESGA Centro de Supercomputación de Galicia. ii, 23, 27, 28, 30

CNN Convolutional Neural Network. 18

DB Base de datos (Data Base por sus siglas en inglés). 31

DeepCoNN Deep Cooperative Neural Networks. 17, 18

DER Dynamic Explainable Recommender. 19, 78

GRU Gated Recurrent Unit. 19

HANN hierarchical attention-based network. 19

IDE Entorno de Desarrollo Integrado (Integrated Development Environment por sus siglas en inglés). 26

INE Instituto Nacional de estadística. 1

JSON JavaScript Object Notation. 31

LDA Latent Dirichlet Allocation. 16

LSTM Long Short-Term Memory. 17, 19

MLP Perceptrón Multicapa (Multi Layer Perceptrón por sus siglas en inglés). 49

NARRE Neural Attentional Regression model with Review-level Explanations. 17–19, 78

NCF Neural Collaborative Filtering. 49

NL Natural Language. 2

NLTK Natural Language Toolkit. 24

NNMF Neural Network Matrix Factorization. 49

PV-DBOW Distributed Bag of Words version of Paragraph Vector. 40

PV-DM Distributed Memory version of Paragraph Vector. 40

RS Sistema de Recomendación (Recommender System por sus siglas en inglés). 1

SGD Descenso de gradientes estocástico (Stochastic gradient descent por sus siglas en inglés). 68

SLURM Simple Linux Utility for Resources Management. 27

TIC Tecnologías de la Información y de las Comunicaciones. 33

TPE Tree-structured Parzen Estimator. 46

Glosario

análisis de sentimiento Clasificación masiva de documentos de manera automática, en función de la connotación positiva o negativa del lenguaje ocupado en el documento, basada en métodos estadísticos. 15, 16

Bag-of-words Método de procesado de lenguaje para representar documentos ignorando el orden de las palabras. 3

corpus Conjunto de frases. 37

Gram Grupo de n palabras, siendo n el tamaño de ventana alrededor de la palabra objetivo que se incluye en el grupo. 37

Mini-batch Cada subdivisión de tamaño pequeño de los datos de entrada de un modelo de aprendizaje máquina. Después de introducir cada Mini-batch, se actualizan los pesos del modelo. 37

palabras vacías Nombre que reciben las palabras sin significado como artículos, pronombres, preposiciones, etc. 4

Paragraph-vector Representación de un documento en un espacio vectorial denso. 40

regresión Conjunto de procesos estadísticos para estimar relaciones entre variables. 17

Skip-gram Grupo de palabras del contexto de la palabra central de un Gram escogido aleatoriamente. 36

Word-vector Representación de una palabra en un espacio vectorial denso. 17

Bibliografía

[1] A. Gastesi, "Imagen e-commerce en españa," 2019. [En línea]. Disponible en: https://www.lavanguardia.com/economia/20190209/46299305726/ecommerce-venta-online-datos-tiendas-fisicas.html

[2] Scrum.org, "Scrum framework image," 2019. [En línea]. Disponible en: https://www.scrum.org/resources/scrum-framework-poster

[3] M. DelSole, "Label encoding vs one-hot encoding image," 2018. [En línea]. Disponible en: https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179

[4] P. Sought, "The cat sat on the mat one hot encoding image," 2018. [En línea]. Disponible en: http://www.programmersought.com/article/6583572078/

[5] H. Kung-Hsiang, "Dense vector similarity image," 2018. [En línea]. Disponible en: https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c

[6] L. Barazza, "Skip grams representation image," 2017. [En línea]. Disponible en: https://becominghuman.ai/how-does-word2vecs-skip-gram-work-f92e0525def4

[7] ——, "Neural network for the skip-gram model image," 2017. [En línea]. Disponible en: https://becominghuman.ai/how-does-word2vecs-skip-gram-work-f92e0525def4

[8] A. Popov, "Neural network models for word sense disambiguation: An overview," *Cybernetics and Information Technologies*, vol. 18, pp. 139–151, 03 2018.

[9] A. K. Samala, "Hierarchical softmax tree like structure image," 2017. [En línea].

Disponible en: https://becominghuman.ai/hierarchical-softmax-as-output-activation-function-in-neural-network-1d19089c4f49

[10] G. Shperber, "Pv-dbow and pv-dm images," 2017. [En línea]. Disponible en: https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e

[11] J. Le, "Recommendation system series part 4: The 7 variants of matrix factorization for collaborative filtering," 2020. [En línea]. Disponible en: https://towardsdatascience.com/recsys-series-part-4-the-7-variants-of-matrix-factorization-for-collaborative-filtering-368754e4fab5

[12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," 2017.

[13] INE, "España en cifras 2018," 2018. [En línea]. Disponible en: https://www.ine.es/prodyser/espa_cifras/2018/26/

[14] C. W. Leung, S. C. Chan, and F.-l. Chung, "Integrating collaborative filtering and sentiment analysis: A rating inference approach," in *Proceedings of the ECAI 2006 workshop on recommender systems*, 2006, pp. 62–66.

[15] H. Wang, N. Wang, and D.-Y. Yeung, "Collaborative deep learning for recommender systems," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2015, pp. 1235–1244.

[16] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 425–434.

[17] J. Gaillard, M. El Bèze, E. Altman, and E. Ethis, "Well-argued recommendation: adaptive models based on words in recommender systems," in *EMNLP-Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1943–1947.

[18] J. Ni, Z. C. Lipton, S. Vikram, and J. McAuley, "Estimating reactions and recommending products with generative models of reviews," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2017, pp. 783–791.

[19] J. Sutherland and K. Schwaber, "The scrum guide," 1993. [En línea]. Disponible en: https://www.scrumguides.org/scrum-guide.html

[20] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," *arXiv preprint arXiv:1607.05368*, 2016.

[21] J. Chai, V. Horvath, N. Nicolov, M. Stys, N. Kambhatla, W. Zadrozny, and P. Melville, "Natural language assistant: A dialog system for online product recommendation," *AI Magazine*, vol. 23, no. 2, pp. 63–63, 2002.

[22] M. Fleischman and E. Hovy, "Recommendations without user preferences: A natural language processing approach," in *Proceedings of the 8th International Conference on Intelligent User Interfaces*, ser. IUI '03. New York, NY, USA: ACM, 2003, pp. 242–244. [En línea]. Disponible en: http://doi.acm.org/10.1145/604045.604087

[23] N. Jakob, S. H. Weber, M. C. Müller, and I. Gurevych, "Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations," in *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*. ACM, 2009, pp. 57–64.

[24] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 2013, pp. 165–172.

[25] G. Ling, M. R. Lyu, and I. King, "Ratings meet reviews, a combined approach to recommend," in *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 2014, pp. 105–112.

[26] Y. Bao, H. Fang, and J. Zhang, "Topicmf: Simultaneously exploiting ratings and reviews for recommendation," in *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv e-prints*, p. arXiv:1301.3781, Jan 2013.

[28] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," *arXiv e-prints*, p. arXiv:1405.4053, May 2014.

[29] R. Catherine and W. Cohen, "Transnets: Learning to transform for recommendation," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 2017, pp. 288–296.

[30] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1583–1592.

[31] D. Cong, Y. Zhao, B. Qin, Y. Han, M. Zhang, A. Liu, and N. Chen, "Hierarchical attention based neural network for explainable recommendation," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, 2019, pp. 373–381.

[32] X. Chen, Y. Zhang, and Z. Qin, "Dynamic explainable recommendation based on neural attentive models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 53–60.

[33] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *IEEE Internet computing*, vol. 7, no. 1, pp. 76–80, 2003.

[34] B. Smith and G. Linden, "Two decades of recommender systems at amazon. com," *Ieee internet computing*, vol. 21, no. 3, pp. 12–18, 2017.

[35] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ser. RecSys '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 191–198. [En línea]. Disponible en: https://doi.org/10.1145/2959100.2959190

[36] E. de Youtube, "[youtube recommendations] ask us anything! youtube team will be here friday february 8th." 2019. [En línea]. Disponible en: https://support.google.com/youtube/thread/1456096?hl=en

[37] G. van Rossum, "Python," 1991. [En línea]. Disponible en: https://www.python.org/about/

[38] O. Travis, "Numpy," 1995. [En línea]. Disponible en: https://www.numpy.org/

[39] E. Rose and B. Bayles, "More itertools," 2012. [En línea]. Disponible en: https://github.com/erikrose/more-itertools

[40] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[41] M. McKerns, "Dill," 2014. [En línea]. Disponible en: https://pypi.org/project/dill/

[42] S. Bird, E. Loper, and E. Klein, "Nltk," 2001. [En línea]. Disponible en: https://www.nltk.org/

[43] R. Řehůřek, "Gensim," 2009. [En línea]. Disponible en: https://radimrehurek.com/gensim/

[44] F. Chollet, "Keras," 2015. [En línea]. Disponible en: https://keras.io/

[45] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in science conference*. Citeseer, 2013, pp. 13–20.

[46] J. Bergstra, "Hyperopt," 2011. [En línea]. Disponible en: https://hyperopt.github.io/hyperopt/

[47] Vooban, "Hyperopt-keras-cnn-cifar-100," 2018. [En línea]. Disponible en: https://github.com/Vooban/Hyperopt-Keras-CNN-CIFAR-100

[48] J. Brains, "Pycharm," 2010. [En línea]. Disponible en: https://www.jetbrains.com/pycharm/

[49] JGraph, "draw.io." [En línea]. Disponible en: https://github.com/jgraph

[50] O. Timeline, "Office timeline," 2018. [En línea]. Disponible en: https://online.officetimeline.com

[51] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2003, pp. 44–60.

[52] U. of Illinois, "Review data sets for "latent aspect rating analysis"," 2011. [En línea]. Disponible en: http://times.cs.uiuc.edu/%7Ewang296/Data/

[53] A. Sivakumar and R. Gunasundari, "A survey on data preprocessing techniques for bioinformatics and web usage mining," 2017.

[54] S. Vijayarani, M. J. Ilamathi, and M. Nithya, "Preprocessing techniques for text mining-an overview," *International Journal of Computer Science & Communication Networks*, vol. 5, no. 1, pp. 7–16, 2015.

[55] J. Camacho-Collados and M. Taher Pilehvar, "On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis," *arXiv e-prints*, p. arXiv:1707.01780, Jul 2017.

[56] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," *arXiv e-prints*, p. arXiv:1310.4546, Oct 2013.

[57] X. Rong, "Word2vec Parameter Learning Explained," *arXiv e-prints*, p. arXiv:1411.2738, Nov 2014.

[58] A. Singhal, "Modern information retrieval: a brief overview," *BULLETIN OF THE IEEE COMPUTER SOCIETY TECHNICAL COMMITTEE ON DATA ENGINEERING*, vol. 24, p. 2001, 2001.

[59] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," *arXiv e-prints*, p. arXiv:1507.07998, Jul 2015.

[60] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Googlenews vectors," 2013. [En línea]. Disponible en: https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

[61] P. Lison and A. Kutuzov, "Redefining context windows for word embedding models: An experimental study," *arXiv preprint arXiv:1704.05781*, 2017.

[62] P. Lerman, "Fitting segmented regression models by grid search," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 29, no. 1, pp. 77–84, 1980.

[63] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization Techniques IFIP Technical Conference*. Springer, 1975, pp. 400–404.

[64] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[65] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, no. 8, pp. 30–37, 2009.

[66] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, p. 5, 2019.

[67] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359 –

366, 1989. [En línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/0893608089900208

[68] E. Dynomant, S. Darmoni, E. Lejeune, G. Kerdelhué, J.-P. Leroy, V. Lequertier, S. Canu, and J. Grosjean, "Doc2vec on the pubmed corpus: study of a new approach to generate related articles," 11 2019.

[69] D. Masters and C. Luschi, "Revisiting Small Batch Training for Deep Neural Networks," *arXiv e-prints*, p. arXiv:1804.07612, Apr 2018.

[70] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning Activation Functions to Improve Deep Neural Networks," *arXiv e-prints*, p. arXiv:1412.6830, Dec 2014.

[71] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[72] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv e-prints*, p. arXiv:1412.6980, Dec 2014.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[74] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.